Using Support Vector Machines in Anomaly Intrusion Detection

by

Eric M Nyakundi

A Thesis

presented to

The University of Guelph

In partial fulfilment of requirements

for the degree of

Master of Science

in

Computer Science

Guelph, Ontario, Canada

ABSTRACT

Using Support Vector Machines in Anomaly Intrusion Detection

Eric M Nyakundi
University of Guelph, 2015

Advisor:
Dr. Charlie Obimbo

Recent increase in hacks and computer network attacks around the world, including Sony Pictures (2014), Home Depot (2014), and Target (2014) gives a compelling need to develop better Intrusion Detection and Prevention systems. Network intrusions have become larger and more pervasive in nature. However, most anomaly intrusion detection systems are plagued by large number of false positives thus limiting their use. In this Thesis as a contribution to building better Intrusion Detection Systems, we classify intrusions using Support Vector Machines and perform experiments to determine their performance and compare them to other classifiers e.g naïve-Bayes, multilayer perceptrons on the network intrusion detection classification task. The classifiers are evaluated on the ISCX2012 dataset. The proposed Support Vector Machine classifier achieves 99.1% average detection accuracy which demonstrates better performance compared to the modified gravitational search algorithm (MGSA) neural network which achieved 97.8% accuracy and the multi-objective genetic algorithm (MOGA) multilayer perceptron which achieved 97% average detection accuracy.

Acknowledgments

I am eternally grateful to my supervisor, Dr Charlie Obimbo, for his support, mentoring and expert guidance during the course of my graduate studies and writing of my thesis. Charlie's advice, encouragement and spiritual guidance have had a big role in the successful completion of my studies.

Additionally, I would like to acknowledge my colleague Ben Ferriman, friends: Maureen, Ali, Dan and the rest for the support, inspiration and encouragement.

I would also like to offer my gratitude to Richard, Beatrice and the family for their financial support and being my family away from home. I am forever indebted to you. I would also like to thank my parents, Joseph and Mary, my brothers, and cousins especially Edith for their prayers and encouragement during my studies and for always believing and supporting me in my endeavors.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Internet and the improved computer technologies has revolutionized business and information system trends. The Internet has brought the ability to do business online and improved the availability of information to the masses. Improved computer technology has increased the ability of organizations to hold vast amounts of information and data that is worth a lot of money. There is a heavy reliance on the Internet and computer networks for the day to day activities of people. Services like online banking, online shopping and many more other services require to be safe and protected in order to avert financial losses.

The protection of computer systems and networks is of great importance to keep information safe from hackers. Current victims of hackers and data breaches include Sony Corporation [17], Home depot [34], and Minecraft [24]. A comprehensive list that contain attacks from 2014 and the beginning of 2015 with the entities name, number of records compromised, the kind of organization attacked is contained in appendix A.2.

Data breaches and hacking cause a lot of financial losses to businesses, organizations and institutions. Big corporations are not the only targets of data breaches but also health institutions, governments, and financial firms due to the vast amounts of valuable information that they store like credit card information, Social Security numbers, bank account information. The losses arise from security costs after the fact, loss of customers due to reduced consumer confidence. Most companies withhold the amount of financial losses incurred from such breaches to protect their reputations. The Table of black market prizes for data in appendix A.1 can give a rough idea of the financial losses a company might suffer from data breaches. These kinds of intrusions make intrusion detection a key area in preventing future attacks.

The goal of intrusion detection is to identify, preferably in real time, unauthorized use, misuse and abuse of computer systems by both system insiders and external penetrators [50]. The intrusion detection problem is becoming more challenging due to the great increase in computer networks connectivity, the thriving technology advancement and the ease of finding hackers for hire.

Intrusion detection systems (IDSs) are security systems used to monitor, recognize and report malicious activities or policy violations in computer systems and networks. IDSs are based on the hypothesis that an intruder's behavior will be noticeably different from that of a legitimate user and that many unauthorized actions are detectable. Some of the security violations that would create abnormal patterns of system usage include unauthorized people trying to get into the system, legitimate users doing illegal activities, trojan horses, viruses and denial of service [14].

IDSs use a slightly different classification of attack types. They are Probe, Remote-to-Local(R2L), User-to-Root(U2R) and Denial-of-Service(DoS). A detailed description is provided in Chapter 2. IDSs are generally categorized as Signature-based (Misuse detection) systems, Behavior-based (Anomaly detection) systems or

Hybrid systems.

Most IDSs that are deployed on networks are misuse IDSs because they are robust and have low false alarm rates. However, they suffer from one major shortcoming, they are not able to detect new attacks. Current research is focused on the anomaly detection approach since it can detect new attacks. Anomaly detection approach suffer from high false positive rates that render them impractical to be implemented in live network settings.

The goal is to find a classifier that can accurately detect network intrusions while at the same time reducing the false positive rates and create systems that do not need expert knowledge to create and update signatures rather learn and update themselves. The system should have low false positive rates to make it practical to be deployed in live network environments so as to improve network security. *Accuracy* as used in this Thesis is the overall value of all correctly classified instances i.e both true positives and true negatives.

1.1 Thesis Statement

By using Support Vector Machines, this Thesis aims to classify intrusions accurately while reducing false positive rates. These network intrusion attacks are SSH attacks, L2L attacks, Botnet attacks,and DoS attacks. A detailed description of the attack types is available in Chapter 3.

After completing research into this problem, this Thesis demonstrates that support vector machines (SVM) produce impressive results and is superior to multilayer perceptron (MLP), radial basis function network (RBF-N), and naïve-Bayes (NB) classifier. The support vector machines achieve a high detection rate of 99.1% with low false positive rate of 4.5% in the intrusion classification task on the ISCX2012 dataset. This demonstrates that SVMs can be used successfully as the classifier of choice in the classification module of a network anomaly intrusion detection system.

1.2 Overview of Thesis

The remainder of the Thesis is organized as follows. Chapter 2, background on intrusion detection and classification is presented. We will then define Support Vector Machines. Finally we have a literature review of intrusion detection and intrusion detection datasets. In Chapter 3, we will look at the dataset used and measurement metrics that are used. The main results are also provided including the methodology and implementation. Then in Chapter 4 we analyze the resulting algorithms. The Thesis concludes in Chapter 5 with a summary and directions for future research.

Chapter 2

Background

2.1 Intrusion Detection

Prior to our discussion on intrusion detection, we will define a few terms from the pioneering work of Anderson [5]. *Vulnerability* is a known or suspected flaw in the hardware or software or operation of a system that exposes the system to penetration or accidental disclosure of information. *Penetration* is obtaining unauthorized (undetected) access to files and programs or the control state of computer system. *Attack* is a specific formulation or execution of a plan to carry out a threat. An attack is successful when a penetration occurs. Lastly, an *Intrusion* is a set of actions aimed to compromise the security goals, namely; integrity, confidentiality, or availability of a computing and networking resource.

Intrusion detection systems (IDSs) are security systems used to monitor, recognize, and report malicious activities or policy violations in computer systems and networks. IDSs are based on the hypothesis that an intruder's behavior will be noticeably different from that of a legitimate user and that many unauthorized actions are detectable. Some of the security violations that would create abnormal patterns of system usage include unauthorized users trying to get into the system, legitimate users doing illegal activities, trojan horses, viruses and denial of service [14]. IDSs are generally categorized as Signature-based (Misuse detection) systems, Behavior-based (Anomaly detection) systems, or Hybrid systems.

2.1.1 *Probe Attacks*

Probe is the action an attacker takes to gather information about the system or network. Probe attacks scan for system vulnerabilities which are usually exploited for other attacks in the future [18]. Probe is similar to a fact finding mission where the goal is to collect as much information about the targeted system including the services the system runs.

An example of a probe is port sweep attack. The aim of this attack is to discover any unsecured open ports that can be used to break into the system. Some ports might be left open by mistake by the system administrators making the system vulnerable to attacks. Port sweeping is quite effective on newly deployed untested systems. It has also become easier to carry out probe attacks because the tools to perform these tasks are more available.

2.1.2 *Remote-to-Local Attacks*

Remote-to-Local (R2L) attack occurs when a remote user tries to get local user privileges. This is a privilege escalation attack. The remote user (does not have an account on that system) exploits vulnerability of the system to gain local access to the system. Social engineering is one of the ways used to carry out this attack. The use of common passwords is a huge problem and to counter that many websites recently have a minimum requirement for a strong password that cannot easily be guessed. Another added vulnerability is the use of the same password to log into different accounts or websites.

2.1.3 *User-to-Root Attacks*

User-to-root (U2R) attack occurs when a person with user privileges tries to acquire root user privileges. It might be carried out by an authorized user who has malicious intentions or by an intruder who was able to break into the system either by social engineering, password sniffing, or dictionary attacks. The most common form of this exploit is the buffer overflow attack [31].

2.1.4 *Denial of Service Attacks*

Denial of Service (DoS) attacks attempt to interrupt or degrade a service that a system provides to its intended users. It is the most common type of network intrusion. A Denial of Service (DoS) attack is carried out by a single person or system whereas a Distributed Denial of Service (DDoS) attack is carried out by more than one person or systems (bots). A bot is a Internet robot that performs automated tasks. DoS attacks are successful if they are able to generate shear volume of network traffic than the system can handle.

DDoS has been recently used as a form of protest by hacktivists. They target an entities website and bombard it with so many requests that the service degenerates or gets crippled completely. DoS are often difficult to prevent actively because they usually appear as normal network traffic. The FIFA 2014 worldcup website was a victim of a DDoS attack [49]. DoS attacks can be implemented in several ways including; internet control message protocol (ICMP) flood attack, SYN flood attack, tear drop attacks and peer-to-peer attacks [31].

2.2 Misuse Intrusion Detection

Misuse detection systems use information about known attacks to detect intrusions. Advantages of misuse detection include high confidence in detection, low false positive rates and an unambiguous detailed identification of attack. It is the most adopted approach in commercial systems because it is well understood and for the above advantages. Disadvantages include the inability to detect unknown attacks, need for expert knowledge to create signatures and regular updating of the signature database to include new attacks. Misuse intrusion detection systems are implemented using four techniques, namely; pattern matching techniques, rule based techniques, data mining techniques, and state-based techniques.

2.2.1 *Pattern Matching Techniques*

Pattern matching techniques are commonly deployed on network intrusion detection systems. The attack patterns are usually modeled and then the IDS matches and identifies them. The network IDSs are modeled based on packet headers, packet contents or both of them. Pattern matching is computational expensive since there are new types and varied forms of attacks emerging everyday. Kumar and Spafford [37] proposed a generic pattern matching model that used Colored Petri Nets. SNORT

[60] is an open source IDS that contains a pattern matching module.

2.2.2 *Rule-based Techniques*

It is one of the earliest used techniques for misuse intrusion detection. The intrusion scenarios are encoded as a set of rules, which are then matched to network or host traffic data. Deviations from the matching process are flagged as intrusions. IDES (Intrusion Detection Expert System) [46] is a rule-based system. The IDES model was a result of Dorothy Denning seminal paper [14]. IDES is trained to detect known intrusions, vulnerabilities, and security policies that are site-specific. IDES has the capability of detecting abuse of privileges by authorized users and also masquerading of other users by authorized users. NIDES (Next-generation Intrusion Detection Expert System) [4] is an extension of the IDES system. NIDES is a hybrid system that has both a rule based component and a statistical model component for anomaly detection.

MIDAS (Multics Intrusion Detection and Alerting System) [63] is an expert system that was developed to perform real time intrusion detection and misuse detection for Dockmaster, the National Computer Security Center (NCSC) networked mainframe system. The rule base is developed in LISP language. The rules in MIDAS are in two separate layers. The first layer (lower layer) fires new events based on a suspicion threshold from matching certain type of user events like number of logins. The second (higher) layer analyses the suspicious events from the lower layer and decides whether the events are suspicious enough to raise an alert [18].

2.2.3 *State-based Techniques*

State-based techniques use system states and state transitions to detect intrusions. They require finite state machine construction for depicting the states and transitions. The states depict the IDS states and transitions characterize events that cause IDS states to change. An automation is flagged as intrusive when it reaches a state that is flagged as intrusive. USTAT (Unix State Transition Analysis Tool) [28] was the first system that proposed this technique.It was developed in UC Santa Barbara. All known intrusion scenarios and vulnerabilities are represented in the form of a state transition diagram. USTAT then monitors the system transition from safe to unsafe using the representations. NETSTAT(Network-based State Transition Analysis Tool) [76] uses the USTAT approach on computer networks.

2.2.4 *Techniques based on Data Mining*

In data mining techniques for misuse intrusion detection the data set is labeled as normal or attack and a learning algorithm is trained using the labeled dataset. Data mining techniques are able retrain intrusion systems automatically based on different types of input data. The data mining models have an edge over pattern based systems by the fact they can be trained automatically which gets rid of the issue of manually creating signatures. There are several different data mining algorithms that have been applied to the misuse detection problem.

Wenke Lee et al [39, 40, 41] proposed a data mining framework. They used a data-centric approach and reported that classification, link analysis (how features are linked), and sequence analysis (sequence of patterns in the system) are useful in the mining of audit data. They experimented on the sendmail system call data and the network tcpdump data using association rules algorithm and frequent episode algorithms.

Data mining based techniques performed well in the 1999 KDD classification cup by classifying known attacks but their performance was fairly poor on new attacks. Data mining techniques require labeled datasets for successful training. Labeling datasets is costly, time consuming, and it is prone to errors. Errors in labeling the datasets causes high false positive rates.

2.3 Anomaly Intrusion Detection

Anomaly intrusion detection systems model the normal behavior of the system, compares it with the current behavior and reports any deviation from the normal behavior. The main advantage is the ability detect unknown attacks. This approach has not been adopted mostly in commercial systems because it works on the assumption that any deviation from the normal behavior is an intrusion which leads to high false positive rates. The high false positive rate is due to the fact that some normal activities can be flagged as anomalous if they deviate from the normal modeled behavior. The current research work that is ongoing is to reduce the high rate of false positives. The ability of anomaly detection systems to correctly classify the four types of attacks DoS, U2R, R2L and probes will greatly reduce the false alarm rates.

Development of an anomaly intrusion detection system consists of two phases: a *training* phase and a *testing* phase. In the training phase, the normal profile model is generated. The generated profile model is applied to new data in the testing phase to evaluate the performance. Different techniques and methods have been proposed for anomaly detection such as statistical models, data-mining based methods, and machine learning based techniques.

2.3.1 *Statistical based Techniques*

Statistical based models observe the activity of subjects and generates profiles to represent their behavior. Profiles include different measures such as activity intensity measure, categorical measures (the distribution of an activity over categories), audit record distribution measure, and ordinal measure (such as CPU usage) [54]. Advantages include it can detect new attacks and can provide specific notification of malicious activities that occur over prolonged time durations. Drawbacks are; it is difficult to determine thresholds that balance the likelihood of false positives with the likelihood of false negatives, they need accurate statistical distributions and not all behaviors can be modeled using pure statistical methods [54]. Some examples of statistical based systems are Haystack [67], and NIDES (Next-generation Intrusion Detection Expert System) [4].

Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD) [57] is a hybrid system used for anomaly and misuse detection. It has two components, a signature analysis component and a statistical based anomaly detection component based on profiles. The statistical anomaly detector labels events as intrusive if they deviate largely from the expected behavior.

2.3.2 *Rule-based Techniques*

Rule based techniques have also been implemented in some anomaly detection models. Wisdom and Sense [75] is an anomaly detection system that has two components: wisdom component and sense component. The wisdom component of the system consists of historical audit data based set of rules that describe the system normal behavior. The wisdom component creates its own set of rules. The sense component analyzes subsequent audit data to determine whether it violates the rule base created by the wisdom component of the system. Other systems include; Network Security Monitor (NSM) [23], Time-based Inductive Machine (TIM) [72] and NADIR (Network Anomaly Detection and Intrusion Reporter) [25].

2.3.3 *Learning Models*

Machine learning based techniques are attracting a lot of research work since they can be used to develop anomaly intrusion detection systems that require less human intervention. This is an attractive capability given that attacks are becoming more complex and varying in nature. Machine learning techniques generate models based on a provided training dataset with instances that are labeled normal or anomalous. Some datasets label the anomalous instances with the specific attack type. For example, probe, DoS, U2R or R2L. The labeling of training datasets used in machine learning is performed manually by human experts which makes it expensive to obtain an accurate labeled dataset.

Machine learning based techniques used in anomaly detection operate in three learning modes; Supervised, semi-supervised and unsupervised techniques. Supervised methods (classification methods), require a training dataset that contains both normal and anomalous labeled instances to generate the predictive model. Semi-supervised methods use a combination of unlabeled data and small amounts of labeled data. This reduces the labeling cost and also harnesses the good performance achieved by supervised methods. Unsupervised learning techniques (clustering methods) do not require training data. Unsupervised approach assumes that most of the network connection instances are normal traffic and that only a very small number of the traffic instances are anomalous. The approach also assumes that there is a statistical difference between the anomalous traffic and the normal traffic [44]. Using the assumptions above, data instances are clustered into groups of similar instances. The cluster with frequent data instances represent normal traffic while cluster with less frequent instances are considered anomalous. Research on machine learning has been carried out using the following techniques; Fuzzy logic [16], Bayes Theory, Support Vector Machine (SVM) [52], Evolutionary computation, Association rules, Clustering and Artificial Neural Networks (ANN) [45].

2.4 Hybrid Systems

Hybrid intrusion detection systems combine both misuse detection and anomaly detection approaches to get the advantages of both approaches. The misuse detection system detects known attacks and anomaly detection approach is utilized for novel or unknown attacks. Hwang et al. [27] proposed to combine a signature based IDS and an anomaly detection system to get the advantages of low false positive rate and also an ability to detect novel attacks. Their anomaly detection system mined anomalous traffic from internet connections and used them to supplement the known signature base of the SNORT [60]. They compared their hybrid detection system with SNORT and BRO [55] systems and achieved 60% positive detection rate compared to 30% and 22% of the SNORT and BRO IDS respectively.

Zhang and Zulkernine [84] proposed a serial hybrid detection system that uses misuse detection method followed by the anomaly detection method. They use random forest technique for the misuse detection system to detect known intrusions and the outliers that result from the random forest algorithm are then utilized by the anomaly detection system. Their hybrid system achieved an overall detection rate of 94.7% and a small false positive rate of 2%.

A novel hybrid system was proposed by Depren et al. [15] that utilized both misuse and anomaly detection approaches. They added a decision system that combined the results of the two approaches. Their anomaly detection system used the Self-organizing Map (SOM) to model user behavior and the misuse detection system used J.48 decision tree for classifying the various attack types. Their experiments showed they achieved better performance from the hybrid system compared to using each of the other systems individually.

Kim et al. [32] proposed a hybrid system that integrated misuse detection system hierarchically with an anomaly detection system. Their misuse detection system is based on the C4.5 decision tree algorithm. The misuse detection system is used to decompose the training dataset for the anomaly detection system into smaller subsets. The anomaly detection model is then created using the one class SVM on every decomposed region of the training subset. They showed that the decomposed training of the anomaly detection model took 50% -60% less time than the conventional models.

2.5 Other Classification

There are a number of other concepts used to classify IDSs [13] as seen on Figure 2.1. *Behavior on detection* describes the response of the IDS during an attack; active systems take active or proactive measures to counter the attack while passive systems merely generate alarms. Most of the IDSs are passive. The few active systems implement measures such as cutting connections that carry attacks, blocking traffic from attacking host, throttling bandwidth, or reconfiguring equipment such as routers or firewalls.

Audit source location distinguishes IDSs based on the kind of information that is analyzed. Host-based IDSs reside on a single system and monitor activity on that machine using audit trails or system logs to detect possible attacks. Research on Host-based intrusion detection include [81, 82]. Network-based IDSs are placed at an important point or points within the network to analyze passing network traffic for signs of intrusion.

Locus of detection describes where the monitoring, detection, and reporting are controlled from. In centralized IDSs, monitoring, detection, and reporting are controlled directly from a central location. In distributed IDSs, monitoring and detection

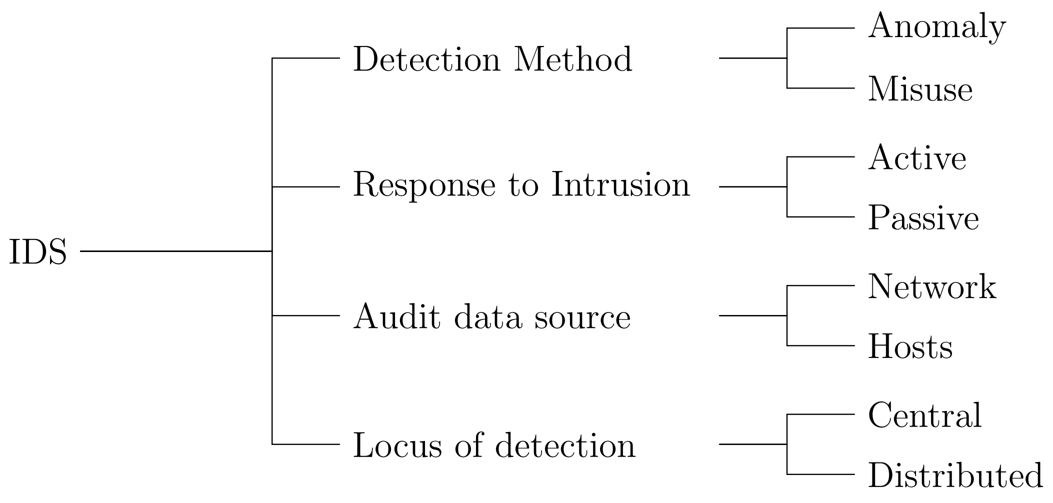


Figure 2.1: Characteristics of Intrusion Detection Systems.

are controlled from a local control node with hierarchical reporting to one or more central location(s). The IDSs can also be off-line or on-line based. Off-line IDSs analyze the data off-line at a later time while the on-line systems analyze the data when the system is still working.

2.6 Support Vector Machines

Support Vector Machines (SVMs) were introduced by Vapnik [11]. SVMs have strong theoretical foundations and they have shown excellent empirical successes in classification tasks such as text classification and digit recognition. SVM separates data into different classes by a hyperplane or hyperplanes since it has the ability to handle multidimensional data. SVMs minimizes empirical classification error and maximizes the margin. It also known as maximum margin classifier.

Suppose we have N training data points$\{(x_1, y_1), \ldots, (x_N, y_N)\}$, where $x_i \in R^d$ and $y_i \in \{+1, -1\}$. The hyperplane equation in d dimensions becomes:

$$(w^T \cdot x) + b = 0 \tag{2.1}$$

where $w \in R^n$ is weight vector, $b \in R$ is a bias value and x is an input vector. The decision function becomes

$$f(x) = sign(w^T \cdot x) + b \tag{2.2}$$

From the structural risk minimization principle, the optimal separating hyperplane of a linear classification is constructed by solving equation (2.3)

$$\text{Min } \frac{1}{2}||w||^2, \tag{2.3}$$

subject to

$$y_i(w^T \cdot x_i + b) \geq 1, \quad i = 1, \ldots, N \tag{2.4}$$

The soft margin SVMs are used to reduce the effects of outliers and mislabeled examples. The method introduces a non-negative slack variable to (2.3) as shown below

$$\text{Min } \frac{1}{2}||w||^2 + C\sum_{i=1}^{N} \xi_i, \tag{2.5}$$

subject to

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \ldots, N \tag{2.6}$$

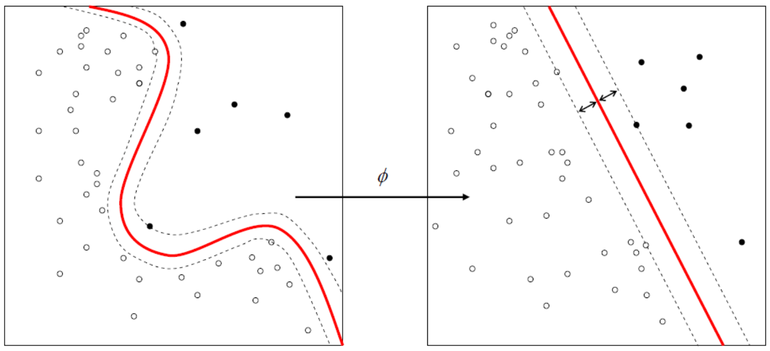


Figure 2.2: Nonlinear Transformation into Higher Dimension Linear Separable Function Using Kernel Trick [78]

where ξ_i is the slack variable and C is a penalty parameter that controls the trade-off between the cost of misclassification error and the classification margin. The dual form of the optimization problem becomes

$$\text{Max} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j), \tag{2.7}$$

subject to

$$\sum_{i=1}^{N} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, N \tag{2.8}$$

where $K(x_i, x_j)$ is the kernel function and α_i are Lagrange Multipliers. The kernel function is one of the important elements attributed to the success of SVMs. The 'kernel trick' which transforms a nonlinear form of SVM to linear form as shown in Figure 2.2 without explicitly computing the products in the high-dimensional feature spaces.

There are three common kernel functions:

Polynomial Kernel function:

$$K(x_i, x_j) = [(x_i \cdot x_j) + 1]^p,$$

where p is the dimension and $p \geq 1$.

RBF Kernel function:

$$K(x_i, x_j) = exp(-\frac{||x - x_i||^2}{\sigma^2}),$$

where σ is the kernel width.

Sigmoid kernel function:

$$K(x_i, x_j) = tanh[v(x_i \cdot x_j) + c]$$

The decision function of the non linear SVM is given by the following function

$$f(x) = sign(\sum_i^N \alpha_i y_i (x_i \cdot x_j) + b) \tag{2.9}$$

Figure 2.3 shows the support vectors, maximum margin, hyperplane and the slack variables.

2.7 Literature Review

2.7.1 Review of Intrusion Detection Literature

Ghosh, Wanken and Charron [19] presented an intrusion detection system that built profiles of software behavior using backpropagation neural network. They showed that neural networks could be used to detect the misuse of programs. They

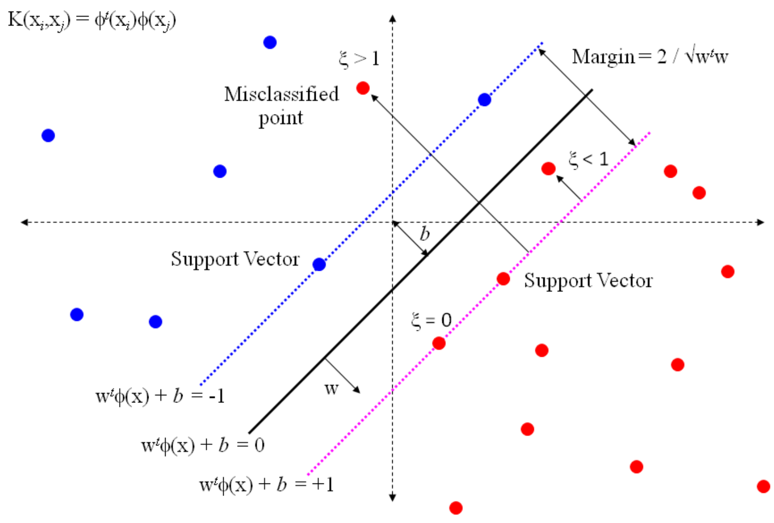


Figure 2.3: Maximum Margin Hyperplanes [3]

trained their neural networks with randomly generated data and this showed the best performance in discovery of novel types of misuse. They demonstrated the advantage of applying the anomaly detection on the process behavior rather than the user behavior. The 5 different neural networks they used showed an average accuracy of between 80.2% and 99.1%.

Lei and Ghorbani [42] used an improved competitive learning neural network (ICLN) for anomaly detection. They compared it to the self organizing maps (SOM). Their proposed algorithm achieved similar accuracy to the SOM but required less computation time during the training phase when tested on the KDDCUP‘99 dataset. Their experiment also confirmed that the ICLN performance was not affected by the number of initial output neurons. Their ICLN achieved an accuracy of 97%. In 2012 [43]they developed a supervised improved competitive learning neural network (SICLN) which was an improvement of the ICLN algorithm. The SICLN achieved higher detection rates(99.66%) compared to SOM (99.62%), ICLN (99.58%) and K-means (99.57%) clustering when tested on the KDDCUP‘99 dataset.

Ghosh and Schwartzbard [20] used a multilayer perceptron (MLP) for anomaly detection in 1999. They used a single hidden layer neural network and also the leaky bucket algorithm in order to provide memory of recent events to the neural networks so as to capture the temporal locality of anomalous events, the events are used for recognizing intrusive behavior. The performance of their proposed model for anomaly detection produced detection accuracy of 77% with 2.2% false alarm rate when tested on the DARPA 1998 dataset. Their misuse detection model produced 90% detection accuracy but a higher false positive rate of 18.7% compared to other misuse detection systems which had the same detection accuracy .

Lee and Heinbuch [38] proposed an IDS made up of hierarchy of neural networks that only monitored selected areas of network behavior (protocols) that are predictable in advance. Their IDS had two level of hierarchy with all packet and queue statistics as the input. Their neural network of choice was the Back Propagation Neural network (BP-NN). Their system was tested on a simulated dataset. When applied to specific areas of the network whose behavior could be predicted *a priori* they achieved up to 100% detection rates.

Zhang, Jiang and Kamel [83] proposed two hierarchical neural networks that used Radial Basis Functions (RBF). The first one was a serial hierarchical intrusion detection system (SHIDS) which updated the structure automatically and adaptively according to how the clustering program identified novel intrusions. They also proposed a parallel hierarchical intrusion detection system (PHIDS) to counter the weakness of the SHIDS. The SHIDS detection errors accumulated since it worked serially and would affect the classifiers downstream and also increase the detection time. The PHIDS would enhance the abilities of the SHIDS by being able to handle the single point failure vulnerability of the SHIDS. They also compared a Back Propagation learner (BPL) with a Radial Basis Functions (RBF). Their results showed the RBF was better with a detection rate of 99.2% and a false positive rate of 1.2% compared to a detection rate of 93.7% and a false positive rate of 7.2% of the BPL. Their SHIDS was able to monitor real-time network traffic and modify its structures adaptively and they discovered that PHIDS runs faster that SHIDS.

In [77], Wang et al proposed use of artificial neural networks and fuzzy clustering in their approach. Firstly, they used the fuzzy clustering technique to generate the training subsets. They then used the different training sets to train different ANNs in the second stage. In the last stage they introduced a fuzzy aggregation meta-learner which they trained using the whole training set. The results of the meta-learner are then combined with the results from the ANNs trained by the subsets and finally they train another ANN from the combined results. Their proposed method achieved a detection accuracy of 96.71% which was slightly better than naïve-Bayes 96.11% and back propagation neural networks 96.65% when tested on the KDD CUP 1999 dataset.

Puttini, Marrakchi and Ludovic [58] introduced a behavior model that uses Bayesian techniques to obtain model parameters with maximal a-posteriori probabilities. The Bayesian technique is used on the detection phase after the behavior model has already been fitted. The Bayesian classification inference is used together with a cluster pertinence inference.

Naïve Bayesian networks are much simpler Bayesian networks where the network is restricted to two layers and it assumes complete independence between the information nodes. Sebyala et al. described a system in [64] that uses a naïve Bayesian network to detect malicious third party executable codes (proxylet) that are used for provision of new services in the current telecommunication infrastructure.

Kruegel et al [35] use a full Bayesian network to overcome the shortfalls of the naïve Bayesian network being identical to a threshold-based system that uses outputs from child nodes in sum computation. Naïve-Bayes also has the restrictions of having a single parent node which complicates the incorporation of additional information. The use of a full Bayesian network allows them to model inter-model dependencies and integrate additional data in order to improve the decision making of the classifier.

Barbara, Ningning and Sushil [7] proposed a pseudo-Bayes estimator that was based on a naïve-Bayes probabilistic model to reduce the false alarm rate on an anomaly detection system called Audit Data Analysis and Mining (ADAM) that they had developed at Center for Secure Information Systems of George Mason University. Mining association rules techniques were used on the network traffic data to detect abnormal events. Their naïve-Bayes classifier was used on the detected abnormal traffic to further classify them into normal instances that had already been encountered by ADAM, known attacks (attacks contained in training dataset) and new attacks. This filtering reduced the number of false alarms greatly. They used the pseudo-Bayes estimators to derive the prior and posterior probabilities of new attacks based on normal instances and known attacks information.

In 2004, Steven Scott [62] proposed a Bayesian paradigm for designing IDS for network systems. He proposed the intrusion detection problem should be modeled stochastically and the models should be split into hierarchical components. He proposed combining competing intrusion detection approaches such as anomaly detection with pattern recognition. The fact that Bayesian methods present evidence of intrusion as probabilities makes it easier to be understood and interpreted by the system administrators who the IDSs report to.

In [33], Koc et al. augmented the naïve-Bayes and structurally extended naïve-Bayes methods with the leading discretization and feature selection methods in order to improve the accuracy and reduce the resources required in the intrusion detection problem. Their results which used the KDD99 dataset, showed that the hidden naïve-Bayes multi-class classification model augmented with various discretization and feature selection methods showed better results in the accuracy of detection, error rate and misclassification cost than the traditional naïve-Bayes model,which was the leading Bayes model in the KDD99 cup winner.

Sheikhan and Jadidi [65] used a modified gravitational search algorithm (MGSA) as a heuristic to optimize a neural anomaly detector. They demonstrated that their proposed approach was able to monitor abnormal traffic flows with an accuracy of 97.8% on the ISCX2012 dataset. They used different metrics to calculate the accuracy. They compared their proposed approach to a particle swarm optimization method.

Kumar and Kumar [36] proposed a multi-objective genetic algorithm (MOGA) approach for effective intrusion detection. The MOGA used had three phases; the first phase entailed designing of a simple chromosome, the second phase obtained ensemble solutions from refining the solutions from phase one. Finally majority voting was used to compute final prediction of the ensemble. The proposed method was evaluated on the KDD Cup 99 dataset and the ISCX2012 dataset. They used a multilayer perceptron (MLP) as the base classifier and they were able to achieve a detection accuracy of 97%.

2.7.2 Literature Review of SVMs in Intrusion Detection

Mukkamala et al [51] were among the first researchers to experiment on anomaly intrusion detection using neural networks and SVMs. They tested the performance of their classifiers on the KDDCUP'99 DARPA dataset [30]. Their classifiers achieved highly accurate results which were greater than 99%. Their SVMs outperformed the neural networks in both training time and detection accuracy. The high detection accuracy might be attributed to the insufficiency of the dataset that they used. In their paper Tavallaee et al. [70] observed that the KDDCUP'99 dataset had redundant records and it is difficult to compare the IDSs that had been evaluated using this dataset. Mukkamala and Sung [69] used the SVMs and neural networks for important feature selection and they still demonstrated high detection accuracies. Mukkamala, Sung and Ribeiro [52] performed evaluations of impact kernels on the accuracy of the SVM classifier in intrusion classification. Their experiments still exhibited high detection accuracy rates similar to the ones mentioned in [51]. They also determined that the ability of SVM classifiers is highly dependent on the kernel type and parameter settings.

In their paper [61], Salem and Stolfo utilize user search behavior to create their model. They hypothesize that each file system is well know by the individual user for them to search in a finite, direct and unique pattern to retrieve information needed to carry out their tasks. However, a masquerader is not expected to know the file system and layout of the system he is masquerading from and would likely have a search pattern that is divergent from the victim or user being impersonated. User activity volume and frequency related to searches and information was audited and used to create the models.

The user behavior models are developed using the one-class support vector machines. The support vector machines (SVMs) is a modern machine learning algorithm that has the best masquerade attack detection accuracy. SVM models are easy to update and are suitable for block-by-block incremental learning. Their focus is on the features that are modeled, limiting number of features chosen so as to reduce sampling time in training data collection.

They developed a new dataset that is more suitable for the masquerade detection problem and they named it RUU dataset and made it publicly available for future research. In their experiment they achieved 100% detection rate with only 1.1% false positives. The use of a small set of features allows for real-time masquerade attack detection since the model requires little system resources. It is also easily deployed since the amount of sampling required is reduced by the fact that profiling is done in a low dimensional space. There is a 74% performance gain when the one class SVM models search behavior compared to the application frequency model that had been used for prior work. The user search behavior model proposed produced far better accuracy compared to prior work that modeled user commands which had high false positive rates with moderate true positive rates.

Nassar et al [53] used SVMs to monitor Voice over Internet Protocol (VoIP) in order to distinguish between attacks and normal activity. Their area of focus was the Session Initiation Protocol (SIP) usually used in Internet telephony. They divided the SIP traffic into small slices, extracted vectors of defined features that characterized each slice, the features were then classified using SVMs. Their experiments demonstrated real time performance and high detection accuracy of flooding attacks and SPAM over Internet Telephony (SPIT). They also compared the performance of different kernels with the Radial Basis Function emerging as the better kernel of choice.

Khan et al [6] presented a way of improving the training time for SVMs especially when used in large datasets using hierarchical clustering analysis. They used the DGSOT algorithm for clustering because it has been shown to overcome the drawbacks of traditional hierarchical clustering algorithms (e.g hierachical agglomerative clustering). The clustering analysis they carried out among the two classes assisted in the discovery of capable boundary points that were used to train the SVM. In order to counter the issue of accuracy degradation of the classifier due to the fact they did not use the original dataset, they repeatedly trained the classifier followed by de-clustering and adding new training examples which are the children of the support vectors.

Chen, Hsu and Shen [9] used SVM and artificial neural networks (ANN) to classify host audit data to detect intrusive program behavior. They chose a frequency-based encoding method over the sequence-based method since it reduces the overhead by checking for attacks at the end of the process rather than after every sequence. Both the SVM with $tf * idf$ (term frequency$*$ inverse document frequency) and ANN with $tf * idf$ achieved 100% accuracy but the SVM had a lower false positive rate of 8.53% compared to 39.25% for the ANN.

Hu, Liao and Vemuri applied Robust SVM (RSVM) [68] to the anomaly detection problem where there was presence of noise in the training data. Noisy data introduces the problem of over-fitting in SVMs. They added noise to the 1998 DARPA BSM dataset [1] and then compared the performance of the RSVM to normal SVMs and k-Nearest Neighbor (kNN). When tested on the clean dataset RSVM and SVM achieved accuracy levels of 81.8% while the kNN achieved 63.6%, the false positive rate was less then 1%. When tested on the noisy dataset RSVM, SVM and kNN achieved accuracy levels of 81.8%, 54.2% and 57.6% respectively with a reported false positive rate of less than 3%. They also showed RSVMs had less support vectors 15 compared to normal SVMs 40 on the noisy dataset.

Horng et al [26] proposed an IDS that combined hierarchical clustering and SVMs. The clustering algorithm was used to preprocess the KDDCUP‘99 dataset [30] before training the SVMs. They chose the Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) as their clustering algorithm. They achieved a detection accuracy of 95.7% and a false positive rate of 0.7%. The detection accuracy was better than the KDD’99 cup winner [56] but the false positive rate was higher.

Table 2.1: Summary of Popular Datasets in the Intrusion Detection Domain [79].

Data source	**Dataset name**	**Abbreviation**
Network traffic	DARPA 1998 TCPDump Files	DARPA98 [1]
	DARPA 1999 TCPDump Files	DARPA99 [2]
	KDD99 Dataset	KDDCUP'99 [30]
	10 KDD99 Dataset	KDD99-10 [30]
	NSL KDD Dataset	NSL-KDD [73]
	Information Security Centre of Excellence 2012 evaluation dataset	ISCX2012 [66]
	Internet Exploration Shootout Dataset	IES [29]
User behavior	UNIX User Dataset	UNIXDS [74]
System call sequences	DARPA 1998 BSM Files	BSM98 [1]
	DARPA 1999 BSM Files	BSM99 [2]
	University of New Mexico Dataset	UNM [10]

2.8 Intrusion Detection Datasets

Datasets used to evaluate the performance of the IDSs is a controversial area with the different challenges associated with datasets. Data used in intrusion detection research work is normally amassed from three sources: network data packets, user input command sequences, or low-level system information, such as system call sequences, log files, and CPU/memory usage. Table 2.1 lists some commonly used benchmark datasets [79]. These are the publicly available datasets that have been used to evaluate either misuse detection or anomaly detection systems. Most IDSs are usually tested using the KDDCUP'99 and DARPA 1998 datasets to test their effectiveness.

In 2000 McHugh [48] did a critique of the 1998 and 1999 DARPA intrusion detection system that had been carried the previous year by Lippmann et al [45] at the Lincoln Laboratory MIT. The two datasets used DARPA98 [1] and DARPA99 [2] were criticized for their shortcomings in as far as how the data was generated. The background data which is normal data free of any attack scenarios was only superficially described and they claimed it was similar to data from Air Force bases. The attack data was also criticized for being synthetic and not realistically distributed in the background noise. The architecture of the network from where the dataset was generated might not be similar to that of a typical Air Force base. He suggested for better traffic characterization and validation of future datasets to handle the dataset shortcomings.

The NSL-KDD dataset [73] addresses some of the problems that were in the KDDCUP'99 dataset. In their paper Tavallaee et al. [70] observed that the KDDCUP'99 [30] dataset had redundant records in the training set which biased the classifiers towards records appearing more frequently. The NSL-KDD train sets contains no duplicate records.The NSL-KDD test sets does not contain duplicate records which removes the possibility of performance bias by learners which have better detection rates on the frequent records. They also reduced the size of the dataset to a reasonable size which would enable researchers to carry experiments on the complete set without need to trim the datasets. This makes it possible for comparison of different research work consistent.

Tavallaee, Stakhanova and Ghorbani [71] in 2010 reviewed the state of experimental practice in the anomaly intrusion detection area and they looked at 276 studies published between the year 2000 and 2008. They found out the most prevalent approach to evaluation of anomaly IDSs was based on fully or partially synthetic datasets. 70% of the studies used publicly available datasets, 32% created their own datasets, 9% used simulation tools and only 7% attempted to test their proposed systems on a real network. They pointed out that privacy was among the main issues in the criticism of existing publicly available datasets since most of them have been sanitized and anonymized to protect privacy. Real traffic on the other hand is usually not guaranteed to be reliable. The other issues pointed out about the data sets include the definition of anomaly, data normalization, feature omission and data reduction.

There are researchers who have opted to create their own datasets to carry out their own research. Salem and Stolfo [61] collected their own dataset and named it RUU (Are you you?) They gathered the data from Windows platform using a host sensor. The gathered data consisted of process name and ID, process path, process parent, process action type (e.g., type of registry access, process creation, process destruction, window title change, etc.), process command arguments, action flags, and registry activity results.

Research is also carried out on datasets that are not available to the public due to privacy issues and or for business secrets. In their paper [47], Mathew et al. use a real Graduate Admission database (University at Buffalo) for their research problem on inside attacker threat against database management systems. Insider attacks occur when authorized users abuse legitimate privileges to masquerade as other users or maliciously harvest data.

Anomaly detectors also use simulated data to test them. Cucurull et al. [12] described an anomaly and mitigation approach for disaster area networks. The mobile ad hoc network (MANET) in their disaster area network was intermittently connected. Intermittently connected MANET (IC-MANET) have contemporaneous routes among the nodes. The focus of their study was on the impact of intrusions on the dissemination protocol. They used a simulation to run their experiments since it should depict a situation of disaster area. In a situation like a disaster dataset collection for mobile ad hoc networks is not possible because of the obvious reasons.

The dataset used in this study is the Information Security Centre of Excellence 2012 intrusion detection evaluation dataset (ISCX2012) [66] from the University of New Brunswick created by Ali Ghorbani and his group. The dataset was chosen because it was generated from a framework that uses guidelines that are important in making a dataset effective in terms of realism, evaluation capabilities, total capture, completeness and malicious activity. The framework that they proposed to generate a benchmark dataset is modifiable, extensible and can be reproduced to reflect the traffic composition and intrusions of a specific time. It is not anonymized due to privacy issues since it was generated in an environment where there is no risk of anyones privacy being compromised. The specifics of the dataset are discussed in the next Chapter.

2.9 The Software Suite

The software suite used is the open source data mining suite Weka from the University of Waikato. The software components used are developed in Java. The machine algorithms packages contained on the suite can be applied directly on a dataset or called from code. Weka was the chosen suite because of its capability of having different preprocessing tools and visualization tools. The multilayer perceptron classifier, naïve Bayes classifier, and RBF network classifier algorithms are available with Weka software suite.

2.9.1 SVM Package

The package used for SVM experiments is the LibSVM tool [8]. The LibSVM gives users the ability to experiment with One-class SVM, Regressing SVM, and nu-SVM. LibSVM has the capability of reporting many useful statistics about the classifier (e.g., confusion matrix,precision, recall, ROC score, etc.).

Chapter 3

Methodology and Implementation

3.1 ISCX2012 Dataset

The dataset used to test the classifiers is the Information Security Centre of Excellence (ISCX 2012) dataset created by Shiravi Ali, Shiravi Hadi, Tavallaee Mahbod and Ghorbani Ali from University of Brunswick ISCX [66]. The dataset was designed specifically for the purpose of developing, testing and evaluation of network intrusion and anomaly detection algorithms. It is among the few datasets that are public and not anonymized for privacy issues. It reflects the current trend of network data. The dataset was generated to address the shortfalls of most of the datasets used in anomaly Intrusion detection [70]. Most of the datasets used to test, evaluate and compare IDSs are internal and cannot be released to other researchers, or are outdated, or suffer from statistical inefficiencies. The dataset contains 17 features and the tag value indicates whether the flow is normal or attack. The features are shown in the Table 3.1

The entire ISCX labeled dataset comprises nearly 1512000 packets with 19 features and collected over seven days of network activity (i.e. normal and intrusion).

Table 3.1: List of Dataset Features.

Attributes	
appName	sourceTCPFlagsDescription
totalSourceBytes	destinationTCPFlagsDescription
totalDestinationBytes	source
totalDestinationPackets	protocolName
totalSourcePackets	sourcePort
sourcePayloadAsBase64	destination
destinationPayloadAsBase64	destinationPort
direction	startDateTime
Tag	stopDateTime

The feature descriptions are explained in Appendix A.3.

3.1.1 Data Preparation

The size of the data means it has to be preprocessed to reduce the size and change the data types for it to work with the chosen algorithm. The dataset is inclusive of a labeled flow file that supports the use of supervised machine learning algorithms. The flow file is labeled with a 'Normal' and 'Attack' tag. The intrusions were carried out on specific days which enables us to rename the attack instances to the specific attack types which are Botnet attacks, Denial of Service attacks, brute force SSH attacks and internal infiltration (L2L) attacks.

The labeled flow is in XML format and it had to be transformed to a format that could be used to train the SVMs. Due to the large size of the dataset we picked 10% of all the labeled data as the test set. The training set is 1% of the entire labeled dataset. The split of 'Normal' and 'Attack' of the whole dataset is as shown on Table 3.3 the chosen from the specific days that had attack files. The days that no attack did not have a labeled data. The normal class data was chosen randomly from the labeled dataset. High frequency attacks like L2L and SSH were also randomly chosen. Due to the low frequency of Botnet and DoS attacks in the dataset all of them were picked from the dataset and occur in both the training and testing dataset.

The L2L attack is an infiltration of the network from the inside of the network. It entails a combination of attacks including a probing attack, a buffer overflow attack and SQL injection. The DoS attack was carried out on the web server to deny HTTP service. The Botnet is a DDoS attack and it was carried out using an IRC botnet. The SSH labeled attack is a probe attack that brute forces the main server using a dictionary with the goal of acquiring a SSH account. The different attack scenarios are further explained in [66].

Further adjustments had to be made to make the data fit for use. Reduction of the number of attributes from all the possible attributes had to be carried out. The following attributes were chosen for the experiment; Application Name, Total Source Bytes, Total Destination Bytes, Total Destination Packets, Total Source Packets, Direction, source TCP Flags Description, Destination TCP Flags Description, Protocol Name, Source Port, Destination Port, and Tag. Some accumulative or redundant attributes, such as Time Start, Time End, and Base64 format payload were removed.

3.2 Accuracy Measurement Metrics

The accuracy of the classifier is determined by different measures. Some of the measures used are false positives and negative rates, confusion matrix, precision, recall

Table 3.2: Dataset Attributes Statistics.

	Training set	**Test Set**
Normal	1227	12285
L2L	60	605
SSH	46	463
Botnet	3	2
DoS	4	36
Total	1340	13391

Table 3.3: Normal vs Attack Distribution.

Flow	**% Normal**	**% Attack**
37870	100	0
13320	98.45	1.55
27555	92.61	7.39
17138	97.8	2.2
57170	93.46	6.54
52226	100	0
39760	98.7	1.3

and F-Measure and also ROC curves. True positive (TP) is an attack that is correctly classified as an intrusion and true negative (TN) normal traffic correctly classified as normal traffic. False positive (FP) is when normal traffic is classified as an intrusion and false negatives (FN) is when an intrusion is classified as normal traffic as shown in Table 3.4. A classifiers goal is to yield as many TP and TN as possible while reducing the FP and FN [18].

Table 3.4: Intrusion Detection System Classification.

	Actual data class	IDS prediction
True Positive(TP)	Attack	Attack
False Positive(FP)	Normal	Attack
True Negative(TN)	Normal	Normal
False Negative(FN)	Attack	Normal

3.2.1 Confusion Matrix

The Confusion Matrix is an evaluating technique applied to any kind of classification problem. The matrix size is dependent on the number of distinct classes that are to be classified. Its used to compare the information visually about the actual class labels against the predicted class labels [18] from classifier . The confusion matrix displays the four values (i.e TN, TP, FN, FP) in a manner the relationship between them is easily comprehensible as shown in Table 3.5.

Table 3.5: Confusion Matrix Table.

		Predicted class	
		Normal	Abnormal
Actual class	Normal	TN	FP
	Abnormal	FN	TP

The other metrics used are precision and recall which are calculated using TP, FN, FP. F-measure is calculated from precision and recall. ROC curves are used to visualize the relation between the TP and FP rates [18].

3.2.2 Precision

Precision is calculated with respect to the intrusion class. It shows how many intrusions predicted by an IDS are actual intrusions [18, 80]. A practical IDS should aim for high precision. A high precision means false alarms are minimized.

$$Precision = \frac{TP}{TP + FP} \quad where\ precision \in [0, 1]. \tag{3.1}$$

Precision cannot be used as the only metric because it does not express the percentage of predicted intrusions compared to all the intrusions in the present in the whole dataset.

3.2.3 Recall

Recall is a metric that shows the percentage of predicted intrusions versus all intrusions present. An IDS classifier should have a high recall value for it to be practical [80, 18].

$$Recall = \frac{TP}{TP + FN} \quad where\ recall \in [0, 1]. \tag{3.2}$$

Recall too has a disadvantage as it does not take into consideration false alarms so an IDS might have a high recall value and a high false alarm rate.

3.2.4 F- Measure

F-Measure is a metric that gives a better measure of accuracy of an IDS. It uses a combination of precision and recall. It is the harmonic mean of precision and recall [80]. An IDS classifier's F-Measure is desired to be high, which implies high precision and high recall values [18].

$$F - Measure = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} \quad where\ F - Measure \in [0, 1]. \tag{3.3}$$

3.2.5 ROC curves

Receiver Operating Characteristics (ROC) curve is an alternative measure of evaluating a models performance. Signal detection theory is the origin of ROC curves.

ROC curve is a plot of true positive rate (y-axis) against the false positive rate (x-axis) [80]. It is used to visualize the relationship between TP and FP rate as seen in Figure 3.1. Its used to help tune a classifier and also compare two or more different classifier models. When comparing two classifiers the one with a higher area under the curve is the better classifier.

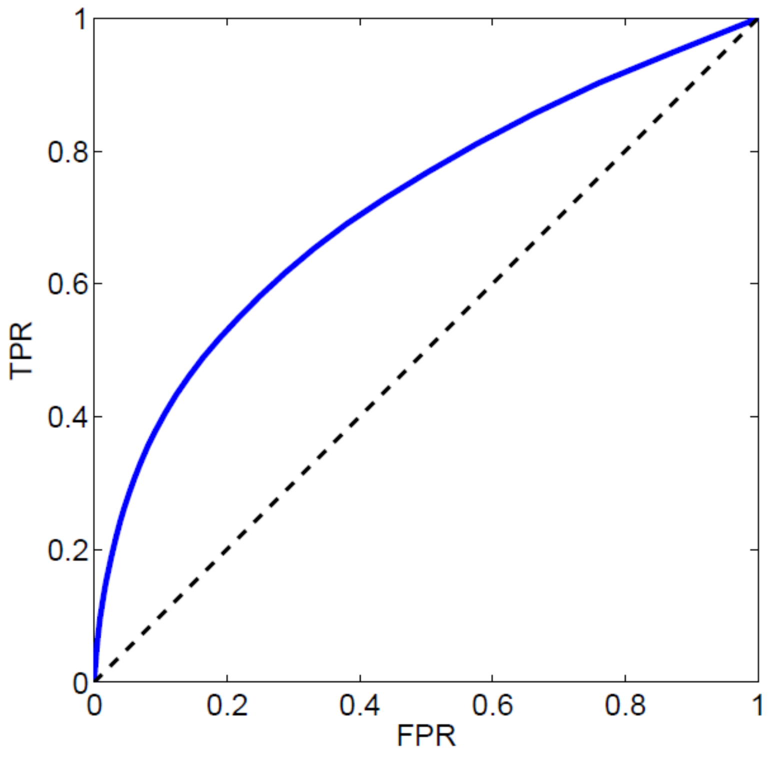


Figure 3.1: ROC Curve Example [78]

3.3 DESCRIPTION OF THE TESTS

Five tests were performed to compare the performance of the Support Vector Machines (SVM) with Multi Layer Perceptron (MLP), naïve-Bayes (NB) and RBF

Network (RBF-N). Two tests were further done to determine the better kernel between the polynomial kernel and RBF kernel. The SVMs were tuned to obtain the best gamma and cost values. The tests were carried on the ISCX2012 data. 1% of the data randomly sampled from the whole data set was used as the training set. The test set contains 10% of the entire data set.

3.3.1 Objective of the Tests

The first test evaluates the performance of the RBF kernel function and polynomial kernel function to determine the better of the two functions when the SVM is applied to the train and test dataset. The SVM that performs better between SVM-RBF and SVM-Poly will be used to compare with the performance of MLP, RBF-N and naïve-Bayes classifier.

3.3.2 General Steps of the Tests

The following steps were followed in the tests that were performed. The training dataset was loaded into the software suite. Depending on the classifier the data was preprocessed accordingly and will be stated in each tests description. The classifier is then applied to the dataset and tuning of the parameters to obtain the best detection rate was carried out. The models are then tested on the testing dataset. All the tests were carried out on the same training and testing dataset.

3.3.3 Test 1: Intrusion Detection Using SVM-Poly

This test is used to demonstrate intrusion detection each class of scenario using SVM classifier that uses a polynomial kernel on a subset of the ISCX dataset. The effect of data normalization is also inspected on the classifier performance.

3.3.4 Test 2: Intrusion Detection Using SVM-RBF

This test is used to demonstrate intrusion detection each class of scenario using SVM classifier that uses a RBF kernel on a subset of the ISCX2012 labeled dataset. The effect of data normalization is also inspected.

3.3.5 Test 3: Intrusion Detection Using MLP

This test is used to demonstrate intrusion detection each class of scenario using Multi Layer Perceptron (MLP) classifier on a subset of the ISCX dataset and the performance compared to the SVM. The effect of data normalization is also inspected on the classifier performance.

3.3.6 Test 4: Intrusion Detection Using RBF-N

This test is used to demonstrate intrusion detection each class of scenario using RBF network classifier on a subset of the ISCX dataset and the performance compared to the SVM. The effect of data normalization is also inspected on the classifier performance.

3.3.7 Test 5: Intrusion Detection Using Naïve-Bayes Classifier

This test is used to demonstrate intrusion detection each class of scenario using naïve-Bayes classifier on a subset of the ISCX dataset and the performance compared to the SVM. The effect of data normalization is also inspected on the classifier performance. Discretization of the dataset will also be investigated.

3.4 PARAMETERS F0R THE TESTS

3.4.1 Parameters for the MultiLayer Perceptron Experiment

A multilayer perceptron neural network requires a number of parameters. There are no strict rules for setting this parameters. A working combination of these parameters were selected based on different tests that were made to test their effect on the classifiers performance. Figure 3.2 shows the layers of the network used in this test. The parameters are listed below.

Input Nodes

The input is divided into the 11 attributes present in the dataset which are: appName, totalSourceBytes, totalDestinationBytes, totalDestinationPackets, totalSourcePackets, direction, sourceTCPFlagsDescription, destinationTCPFlagsDescription, protocolName, sourcePort and destinationPort. The Tag attribute is the only used as a label.

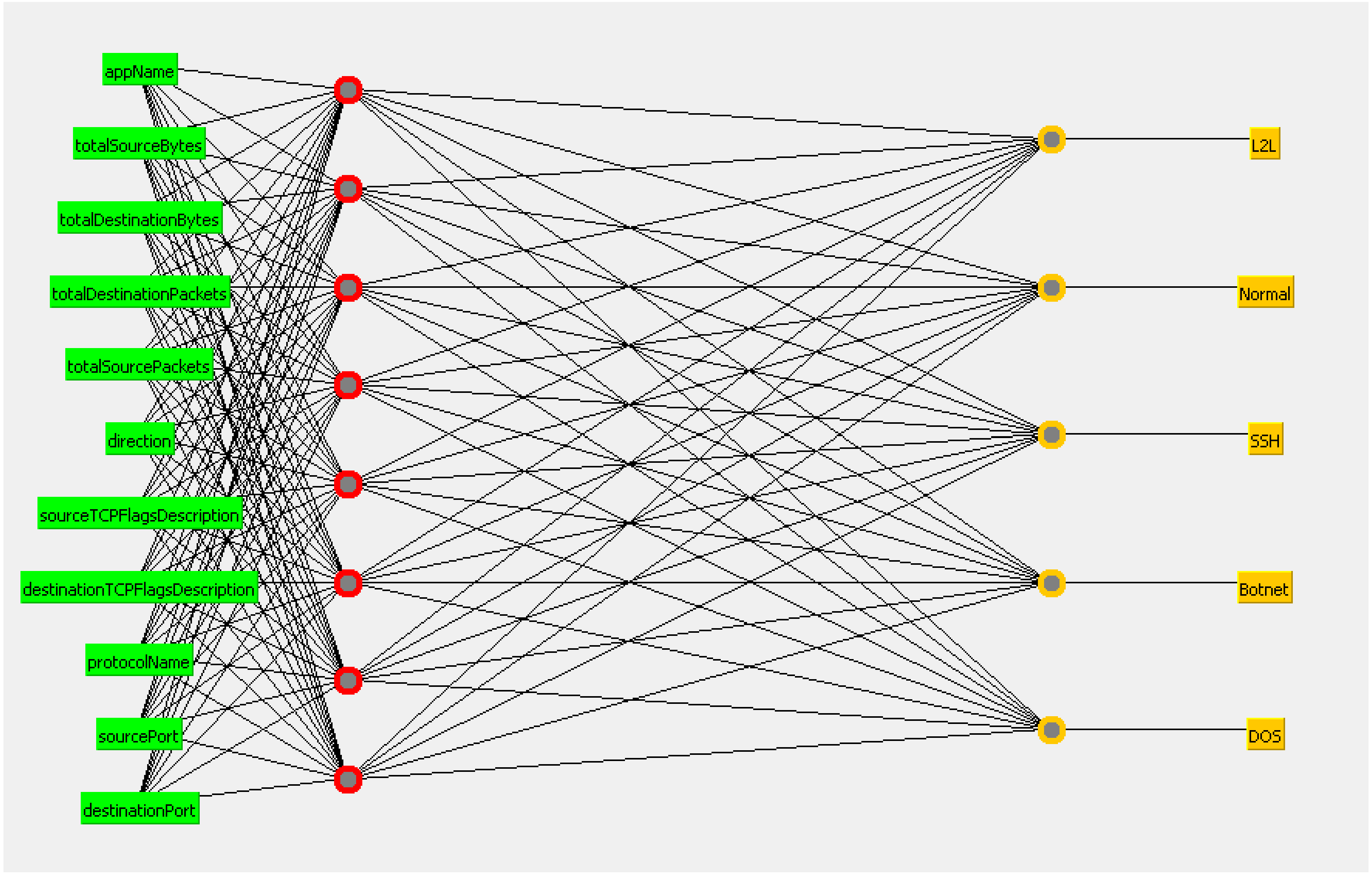


Figure 3.2: The Multilayer Perceptron Network

Momentum

This parameter is used for speeding up the learning rate. When several consecutive input patterns have the same general bias momentum value allows the neural net to have large weight adjustments that are reasonable. It also prevents a large oscillation from any one single training pattern. It speeds up the learning rate as long as the error rate is decreasing. This study used a momentum rate of 0.2.

Learning Rate

The learning rate parameter is used to determine the rate at which the weight values of the layers are modified. A higher learning rate translates to faster network training. However, if the learning rate is too high the network might become unstable [22]. We use a learning rate of 0.3 in this study.

Output Nodes

The output is five nodes which represent the different classes, namely; Normal, L2L, SSH, Botnet and DoS.

Number of Hidden Nodes

The number of hidden nodes for our study are 8 calculated by the formula given below.

$$HiddenNodes = \frac{(attributes + classes)}{2}. \tag{3.4}$$

3.4.2 Naïve-Bayes Classifier

The naïve-Bayes classifier is based on the Bayes theorem. It operates on a strong independence assumption that the attribute values of the classes are not dependent on the values of other attributes. This assumption is known as the class conditional independence. The assumption, even though unrealistic makes the naïve-Bayes classifier remarkably successful in practice, often competing with other classifiers like decision trees and some selected neural network classifiers in terms of performance.

The following steps explain the basic working principle of the naïve-Bayes classifier:

- Consider a set T, of training samples, having their respective class labels, C_1,....,C_m.
- Let each sample be represented by a vector $X = x_1, ..., x_n$, characterizing n measured values of attributes $A_1, ..., A_n$, respectively.
- Thus if X is a sample from a different set of data, then it would be classified as belonging to C_i iff

$$\max_{c_i} P(C_i|X) \tag{3.5}$$

- We call C_i, the class for which $P(C_i|X)$ is maximized, posterior hypothesis. From Bayes theorem

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}. \tag{3.6}$$

- Only $P(X|C_i)P(C_i)$ is required to be maximized because $P(X)$ is equal for all classes. The classes $P(C_1) = P(C_2) = ... = P(C_k)$ are assumed to be equally likely whenever probabilities are unknown a priori, hence only $P(X|C_i)$ is maximized.

- It is computationally costly to compute $P(X|C_i)$ for a dataset containing many attributes. The computation cost of evaluating $P(X|C_i)P(C_i)$, is reduced by the simple assumption that all attributes are independent. The likelihood can be decomposed into a product of dimension-wise probabilities because of the independence assumption.

$$P(X|C_i) \approx \prod_{k=1}^{n} P(x_k|C_i). \tag{3.7}$$

  The probabilities $P(x_1|C_i), P(x_2|C_i), ..., P(x_n|C_i)$ are approximated from the training set T.

- Every class C_i has its $P(X|C_i)P(C_i)$ evaluated to predict the class label of X. Thus the label of X is C_i iff it is the class maximized by $P(X|C_i)P(C_i)$.

The naïve-Bayes used in this study is from the WEKA software suite [21].

Naïve-Bayes Parameters

There are only two available parameters for the naïve-Bayes implementation in the software suite;

1. Type of estimator. A kernel estimator is preferred for numeric attributes otherwise a normal distribution is used. The normal distribution was used.

2. Supervised discretization was used as it produced better classification accuracy compared to the non-discretized form.

3.4.3 RBF Network

A radial basis function network is an artificial neural network. It uses radial basis functions as the activation functions for the neurons. This test uses a normalized Gaussian RBF network. The k-means clustering algorithm is used to provide the basis functions.

RBF Networks Parameters

There are only two available parameters for the RBF network implementation in the software suite;

1. number of clusters. It determines the number of clusters that will be generated by the k-Means clustering algorithm.

2. minimum standard deviation. Sets the minimum standard deviation for the clusters.

3.4.4 Parameters for the SVM Experiments

The following are the available parameters for the implementation in the software suite.

Kernel Type

It sets the type of kernel that will be used by the SVM. The radial basis function (RBF) and polynomial kernels will be used in the tests.

Cost Function

The cost function is known as the penalty factor. It controls the trade-off between complexity of decision rule and frequency of error [11]. Different values of the cost function were tested. 1.0 was the value chosen for the study.

Gamma

It is the parameter used in the RBF kernel to vary the under fitting and over fitting of the model. A small value of Gamma gives low bias and high variance and large value of gamma has the opposite effect of high bias and low variance. A gamma value of 0.1 was chosen for this study.

Normalize

It determines whether the data will be normalized or not before training. The option that will improve the classifiers performance will be chosen.

Chapter 4

EVALUATION

The primary objective of an Intrusion Detection system is to detect intrusions that occur in the system, while maximizing detection accuracy and reducing the false positive rate. The ideal scenario of a practical Intrusion Detection system classifier is a high precision, a high recall, and a high F-value for all attack types and the entire system and with minimal false positive rates.

The proposed SVM classifier achieves 99.1% detection accuracy when evaluated on the ISCX2012 dataset which shows better performance compared to the modified gravitational search algorithm (MGSA) neural network used in [65] with 97.8% detection accuracy and the multi-objective genetic algorithm (MOGA) multilayer perceptron used by [36] with 97% average detection accuracy.

For all the graphs displayed on this Chapter, the vertical axis represents percentages of the different metrics used to evaluate the different classifiers used. The legends indicate the different metrics; precision, recall and F-value. The horizontal axis varies and will be specified for each graph.

Recall that

1. Precision (P) (page 39) gives the accuracy of intrusion prediction

$$P = \frac{TP}{TP + FP} \quad where\ P \in [0, 1]. \tag{4.1}$$

2. Recall (R) (page 40) gives the percentage of correctly determined intrusions in the specific class

$$R = \frac{TP}{TP + FN} \quad where\ R \in [0, 1]. \tag{4.2}$$

3. F-Value (F) (page 40) is the harmonic mean of P and R

$$F = \frac{2}{\frac{1}{P} + \frac{1}{R}} = \frac{2PR}{P + R} \quad where\ F \in [0, 1]. \tag{4.3}$$

4.1 SVM Algorithm Results

The first simulation run was carried out to find the performance of the SVM classifier. Both the polynomial kernel and the radial basis function kernel were tested. Table 4.1 shows the performance of the two kernels. There is a small difference in performance in all the 3 metrics. The metrics used for comparison are defined in Chapter 3.2.

The average performance of the SVM with polynomial kernel (SVM-P) is better than the average performance of the SVM with radial basis function kernel (SVM-RBF). All the other classifiers performances will be compared to the SVM-P classifier in this study. SVM-P had an average precision, an average recall, and an average F-value of 99.1% across all classes compared to an average precision of 98%, an average recall of 98.2%, and an average F-value of 98.1% across all classes. The individual results are discussed in the following sections.

Table 4.1: SVM-P and SVM-RBF Performance Comparison.

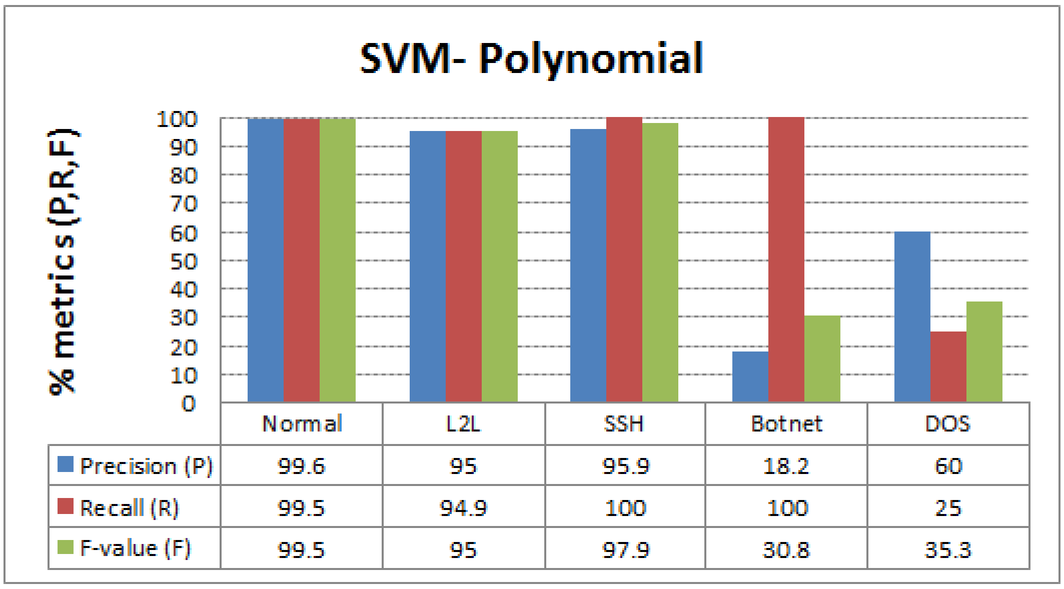
	SVM-P	SVM-RBF
Precision	99.1	98
Recall	99.1	98.2
F-value	99.1	98.1

4.1.1 SVM-Polynomial Algorithm Results

After reviewing results the SVM with polynomial kernel (SVM-P) is found to be the better performing kernel. The process of tuning the classifier is carried out. Different values of the cost function were tested and a value of 1.0 was chosen. Normalization of the data sometimes improves the performance of some classifiers. Normalization was found not to have an effect on the SVM-P classifier.

Support vector machines with the polynomial kernel achieved an average precision value of 99.1%, an average recall value of 99.1%, and an average F-value of 99.1% over all the classes of the dataset. Normalization of the data does not affect the performance of the classifier. The classifier achieves high precision, recall and F-value for high frequency classes normal, L2L and SSH. It performs poorly on the low frequency botnet and DoS classes as seen in Figure 4.1. The Botnet class has a precision of 18.2% but a recall of 100%. The reason why the recall for the Botnet class is high is because the formula of calculating recall does not take into account the false positives (i.e 9 normal instances were predicted as Botnet attacks) as seen on the confusion matrix Table 4.2 (Confusion matrix table was defined on page 39).

The DoS class achieves a precision of 60% and a recall of 25%. A high number of DoS instances were misclassified as Normal, L2L, and SSH instances.



	Normal	L2L	SSH	Botnet	DOS
🟦 Precision (P)	99.6	95	95.9	18.2	60
🟥 Recall (R)	99.5	94.9	100	100	25
🟩 F-value (F)	99.5	95	97.9	30.8	35.3

Figure 4.1: SVM Polynomial Results

Table 4.2: SVM-P Confusion Matrix Table.

		Predicted class				
		Normal	L2L	SSH	Botnet	DoS
Actual class	Normal	12224	27	19	9	6
	L2L	31	574	0	0	0
	SSH	0	0	463	0	0
	Botnet	0	0	0	2	0
	DoS	23	3	1	0	9

4.1.2 SVM-RBF Algorithm Results

Support vector machine with the radial basis function kernel (SVM-RBF) is also implemented and all the parameters tuned. The same cost function of 1.0 chosen for the SVM-P is also used here. Different values of gamma are also tested to vary the effect of under-fitting and over-fitting. A gamma value of 0.1 is used.

SVM-RBF with normalization achieved an average precision value of 98%, an average recall value of 98.2%, and an average F-value of 98.1% over all the classes of the dataset. Without normalization it achieved an average precision value of 93.3%, an average recall value of 93.1%, and an average F-value of 90.7% over all the classes of the dataset. Normalization of the data increases precision, recall and F-value by 4.7%, 5.1% and 8.4% respectively. The SVM-RBF classifier has no precision, recall and F-value for the low frequency DoS class of attacks.

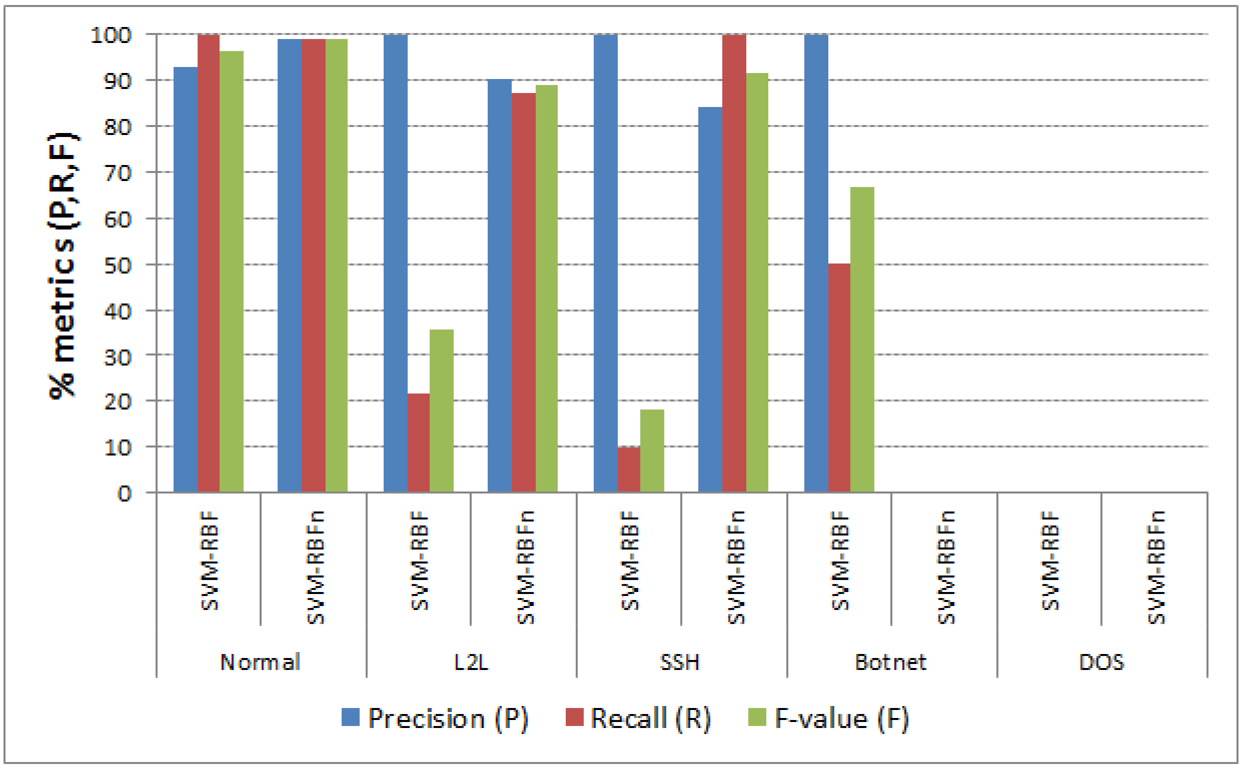


Figure 4.2: Comparison of SVM-RBF and SVM-RBF Normalized

The results from the normalization of the data shows increase in the performance of the high frequency classes normal, L2L and SSH as seen on Figure 4.3. Normalization has an effect on the low frequency Botnet class, the performance gets worse for the class from 66.7% to 0%. As observed before, the normalization has no effect on the Botnet class as the classifier is still unable to detect the DoS attack.

Table 4.3: SVM-RBF Confusion Matrix Table.

		Predicted class				
		Normal	L2L	SSH	Botnet	DoS
Actual class	Normal	12164	37	84	0	0
	L2L	527	78	0	0	0
	SSH	0	0	463	0	0
	Botnet	0	0	2	0	0
	DOS	17	18	1	0	0

4.1.3 SVM-P Comparison to SVM-RBF Results

After conducting the simulations it is evident that the SVM with polynomial kernel (SVM-P) has better F-values than support vector machine with the radial basis function kernel (SVM-RBF) on all classes of the data. SVM-RBF does not detect the low frequency classes of Botnet and DoS as seen on Table 4.3. The polynomial kernel becomes the clear kernel choice and will be utilized for the rest of the experiments of the support vector machines.

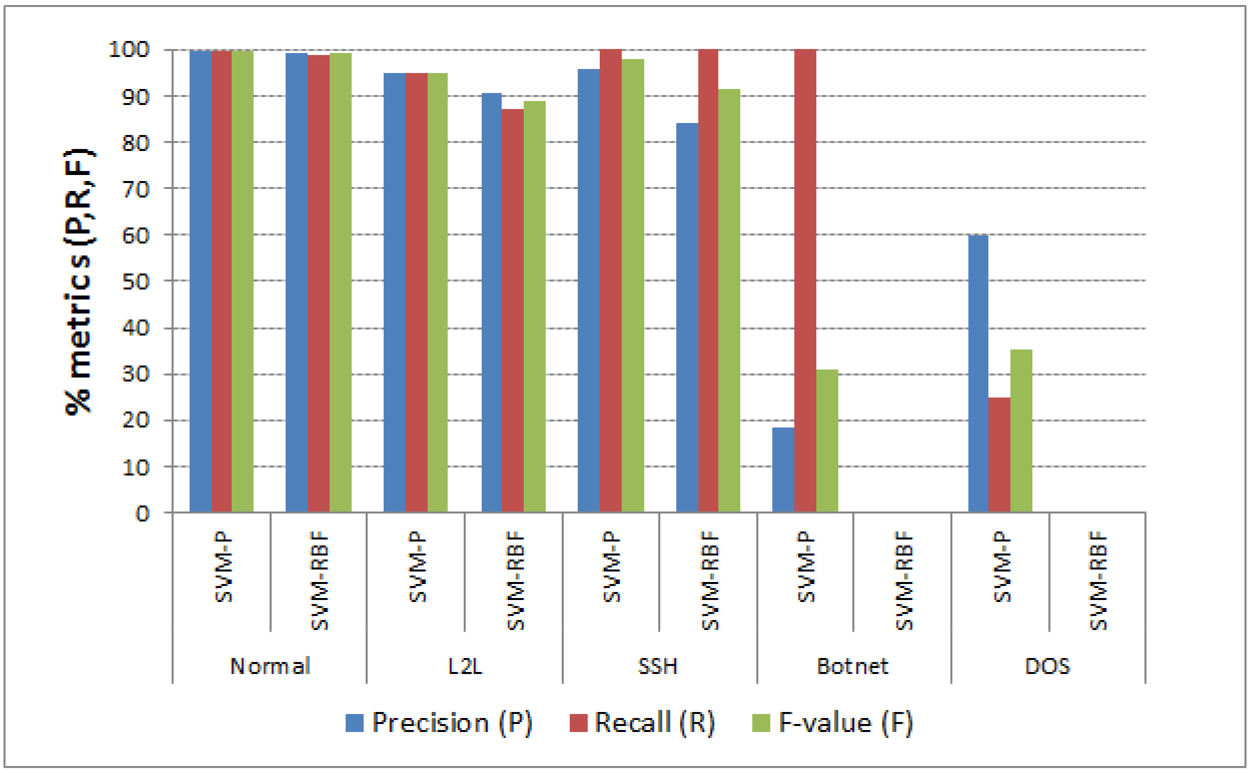
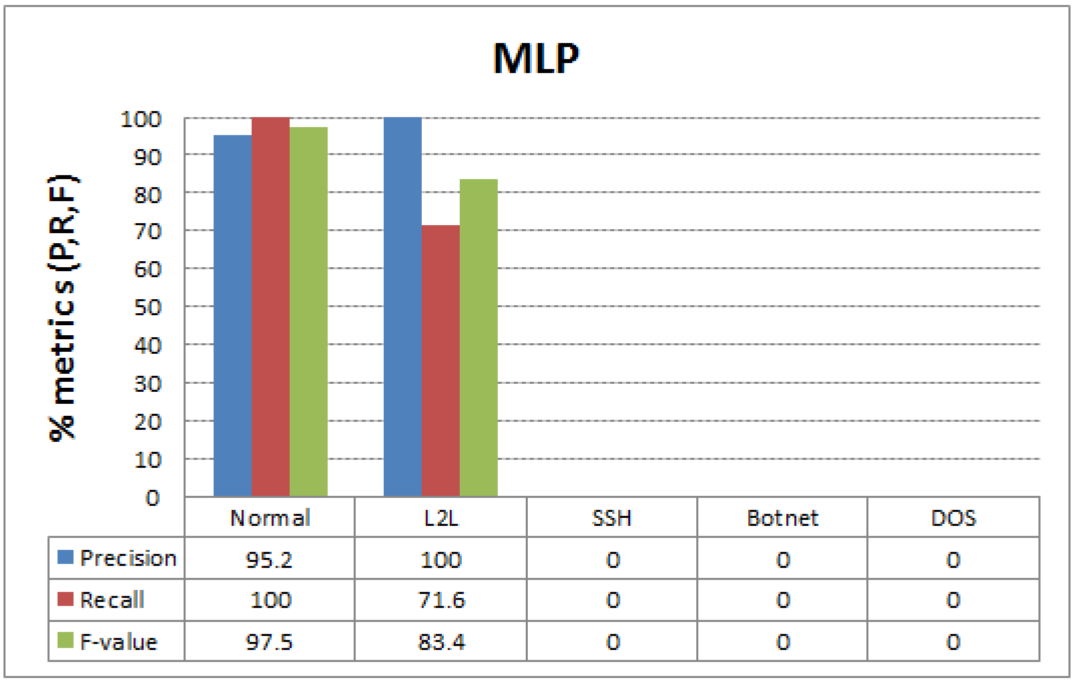


Figure 4.3: Comparison of SVM-P and SVM-RBF

4.2 MLP Algorithm Results

The goal of implementing this algorithm is to do a comparative study on the detection accuracy between the multilayer perceptron (MLP) classifier and our choice classifier (SVM). The classifier is initially trained on the dataset to get the base results. The parameters are then adjusted to achieve the best intrusion detection accuracy. A momentum rate of 0.2 was found to be optimum. The learning rate of 0.3 is also used for the training. The MLP uses 8 hidden nodes which are calculated as shown in Equation 3.4. The dataset is normalized.

The MLP classifier is only able to classify the high frequency Normal and L2L class as seen on Figure 4.4. Normal class had a precision of 95.2%, a recall of 100%, and an F-Value of 97.5%. L2L class had a precision of 100%, a recall of 71.6% and an F-Value of 83.4%. All the other classes were misclassified as seen on the confusion matrix on Table 4.4.



	Normal	L2L	SSH	Botnet	DOS
🟦 Precision	95.2	100	0	0	0
🟥 Recall	100	71.6	0	0	0
🟩 F-value	97.5	83.4	0	0	0

Figure 4.4: MLP Results

Table 4.4: MLP Confusion Matrix Table.

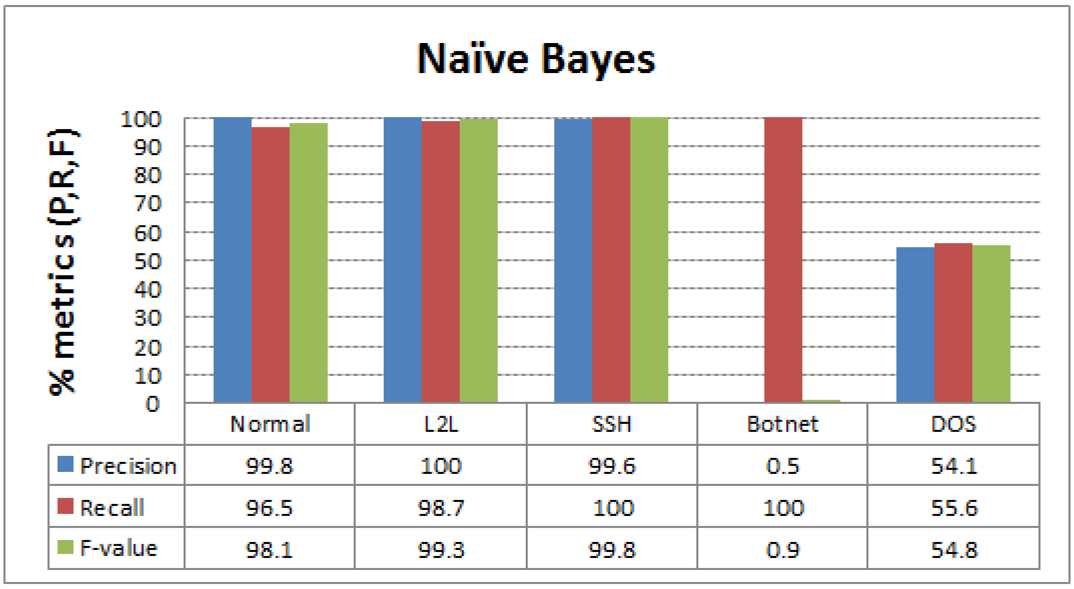
		Predicted class				
		Normal	L2L	SSH	Botnet	DoS
Actual class	Normal	12281	0	2	0	2
	L2L	433	114	0	0	58
	SSH	0	0	463	0	0
	Botnet	0	2	0	0	0
	DoS	0	36	0	0	0

4.3 Naïve-Bayes Algorithm Results

In this simulation, the naïve-Bayes classifier is first trained to determine the base results on the provided dataset. It achieves a detection accuracy of 94.1% and a false positive rate of 35%. The classifier is tuned and the data is discretized to handle continuous data. The performance improves to 96.6% detection accuracy and 1.5% false positive rate. That is an improvement of 33.5% on the false positive rate showing that the naïve-Bayes classifier performs better on discretized data.

The results show that the naïve-Bayes classifier performed well on the high frequency Normal, L2L and SSH classes as seen on Figure 4.5. Normal class has a precision of 99.8%, a recall of 96.5%, and an F-Value of 98.1%. L2L class has a precision of 100%, a recall of 98.7%, and an F-Value of 99.3%. SSH class has a precision of 99.6%, a recall of 100% and an F-Value of 99.8%. The naïve-Bayes classifier has a low precision value for the Botnet class of 0.5% but a high recall value of 100%.

There is an interesting observation when it comes to the low frequency class DoS. The classifier achieved a precision of 54.1%, a recall of 55.6%, and an F-Value of 54.8%. This is the best performance of this low class even though it was not an ideal result. The confusion matrix on Table 4.5 shows how many attributes were correctly and incorrectly classified.



	Normal	L2L	SSH	Botnet	DOS
🟦 Precision	99.8	100	99.6	0.5	54.1
🟥 Recall	96.5	98.7	100	100	55.6
🟩 F-value	98.1	99.3	99.8	0.9	54.8

Figure 4.5: Naïve-Bayes Results

Table 4.5: NB Confusion Matrix Table.

		Predicted class				
		Normal	L2L	SSH	Botnet	DoS
Actual class	Normal	11853	0	1	416	15
	L2L	4	597	0	2	2
	SSH	0	0	463	0	0
	Botnet	0	0	0	2	0
	DoS	14	0	1	1	20

4.4 RBF Network Algorithm Results

The radial basis function network (RBF-N) is the last algorithm implemented to compare its detection accuracy to our choice classifier (SVM). The classifier is initially trained on the dataset to get the base results. The parameters are then adjusted to achieve the best intrusion detection accuracy. Two clusters are used by the k-means clustering algorithm. A standard deviation value of 0.1 is chosen. The classifier performs very poorly on normalized data. The results discussed below are on non-normalized data.

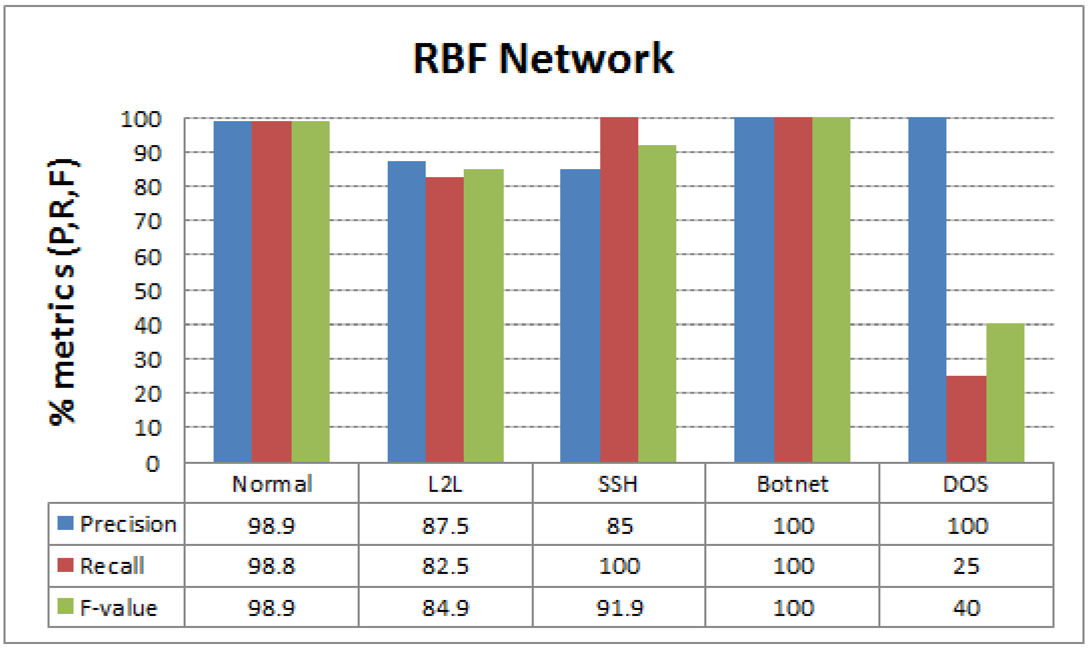
The RBF network classifier performance of the different classes is seen on Figure 4.6. Normal class has a precision of 98.9%, a recall of 98.8%, and an F-Value of 98.9%. SSH class has a precision of 85%, a recall of 100%, and an F-Value of 91.9%.

There are two interesting results observed from this classifiers performance. The L2L class achieves a precision of 87.5%, a recall of 82.5%, and an F-Value of 84.9%. All the other classifiers achieve a relatively high accuracy detection of the high frequency classes except this classifier. Also the RBF network classifier achieved 100% classification on the low frequency class Botnet with a precision, recall and F-Value of 100%.

The low frequency class DoS achieved a precision of 100%, a recall of 25%, and an F-Value of 40%. The confusion matrix on Table 4.6 shows how many attributes were correctly and incorrectly classified by the RBF network classifier.

Table 4.6: RBF-N Confusion Matrix Table.

		Predicted class				
		Normal	L2L	SSH	Botnet	DoS
Actual class	Normal	12134	70	81	0	2
	L2L	106	499	0	0	58
	SSH	0	0	463	0	0
	Botnet	0	0	0	2	0
	DoS	25	1	1	0	9



	Normal	L2L	SSH	Botnet	DOS
🟦 Precision	98.9	87.5	85	100	100
🟥 Recall	98.8	82.5	100	100	25
🟩 F-value	98.9	84.9	91.9	100	40

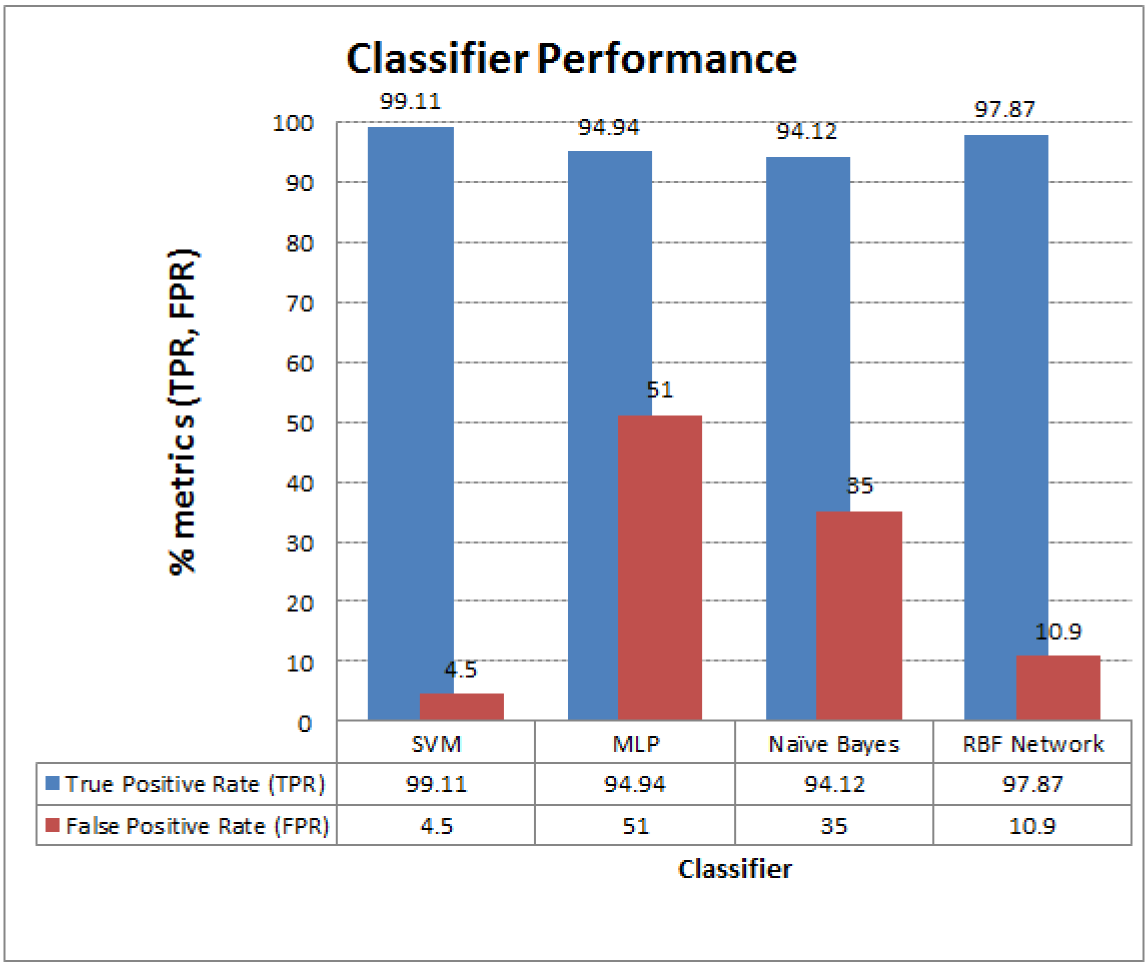
Figure 4.6: RBF Network Results

4.5 Results Comparison

The dataset used contains five different classes; the normal class and four attack type classes. It is important to analyze and understand how each class is handled by the different classifiers used.

Support Vector Machines (SVM-P) with a polynomial kernel produced the highest detection rates compared to the Multilayer Perceptrons (MLP), naïve-Bayes (NB), and radial basis function network (RBF-N). Figure 4.7 shows the average detection rates of the tested classifiers over all the classes. The detection rate of the SVM-P was 99.11%, average detection rate of the MLP was 94.94%, average detection rate using NB was 94.12%, and the average detection rate for the RBF-N was 97.87%.

It is evident that the classifiers perform differently on the different classes. The performance of the classifiers on each class will be compared in the next Section.



	SVM	MLP	Naïve Bayes	RBF Network
🟦 True Positive Rate (TPR)	99.11	94.94	94.12	97.87
🟥 False Positive Rate (FPR)	4.5	51	35	10.9

Figure 4.7: True Positive Detection.

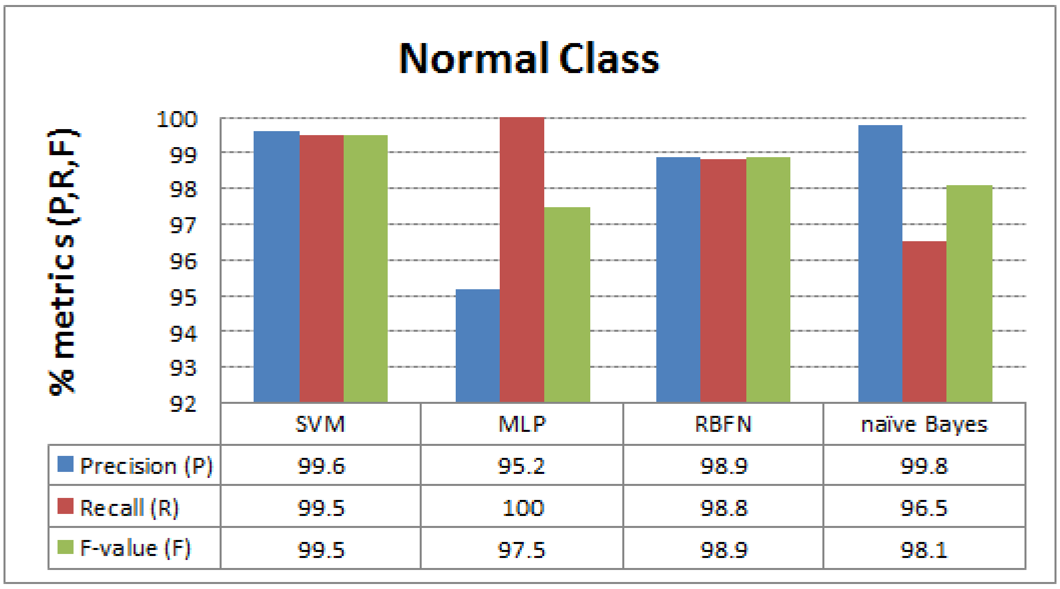
The SVM classifier still performed better than the other classifiers when it came to false positive rates. It achieved an overall false positive rate of 4.5% compared to 10.9%, 35% and 51% of the RBF network, naïve-Bayes and MLP respectively. From observation of the results obtained the SVM classifier is the better classifier overall even though it is outperformed by some of the classifiers on some classes like L2L, SSH, and DoS. It can also be deduced that the classifiers do not perform very well on the low attribute frequency classes like DoS and botnet. The MLP especially performed poorly on the low frequency classes SSH, DoS and Botnet.

4.5.1 *Normal Class*

The normal class contains the normal network flow of the data. The wrong classification of this class as an attack produces a false positive. An IDS that generates a high number of false positives is not practical. The false positive is also used in the calculation of the precision metric Equation 3.1 on page 39.

All the classifiers were able to achieve high detection rates on the normal class. SVM outperformed all the others with a precision value of 99.6%, a recall value of 99.5%, and F-value of 99.5%. These values were even higher than the average precision, average recall, and average F-value of the SVM classifier. The RBF network classifier was the second best with a precision value of 98.9%, a recall value of 98.8%, and F-value of 98.9%. It was followed by naïve-Bayes and then MLP as shown in Figure 4.8.

The normal class has the highest attribute frequency. The normal class is 91% of both the training and testing dataset. The good performance might be attributed to this fact.



	SVM	MLP	RBFN	naïve Bayes
🟦 Precision (P)	99.6	95.2	98.9	99.8
🟥 Recall (R)	99.5	100	98.8	96.5
🟩 F-value (F)	99.5	97.5	98.9	98.1

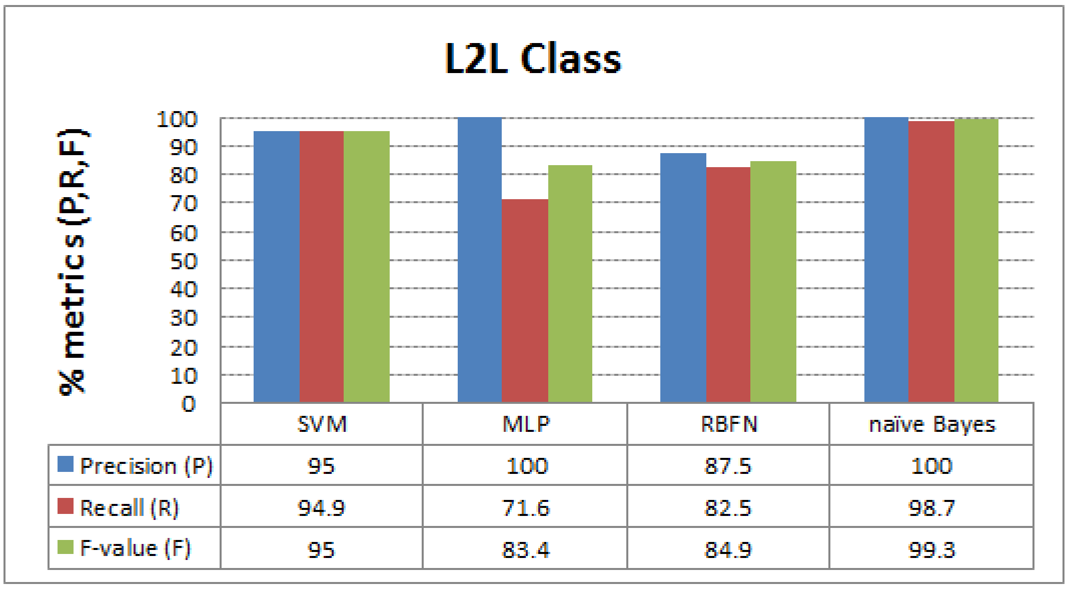
Figure 4.8: Normal Class Performance.

4.5.2 *L2L Class*

The Local-2-Local (L2L) attack type is the most prevalent attack type on the dataset. A wrong classification of this class as normal produces a false negative. High false negatives affect the recall value of the classifier.

In the L2L class the naïve-Bayes classifier outperformed all the other classifiers with a precision value of 100%, a recall value of 98.7%, and F-value of 99.3%. The SVM classifier is the second best with a precision value of 95%, a recall value of 94.9%, and F-value of 95%. The RBF network and the MLP did not perform as well with F-Value of 84.9% and 83.4% as seen on Figure 4.9

The L2L class has the second highest attribute frequency. The training data and the testing data contains 4.5% of L2L class.



	SVM	MLP	RBFN	naïve Bayes
🟦 Precision (P)	95	100	87.5	100
🟥 Recall (R)	94.9	71.6	82.5	98.7
🟩 F-value (F)	95	83.4	84.9	99.3

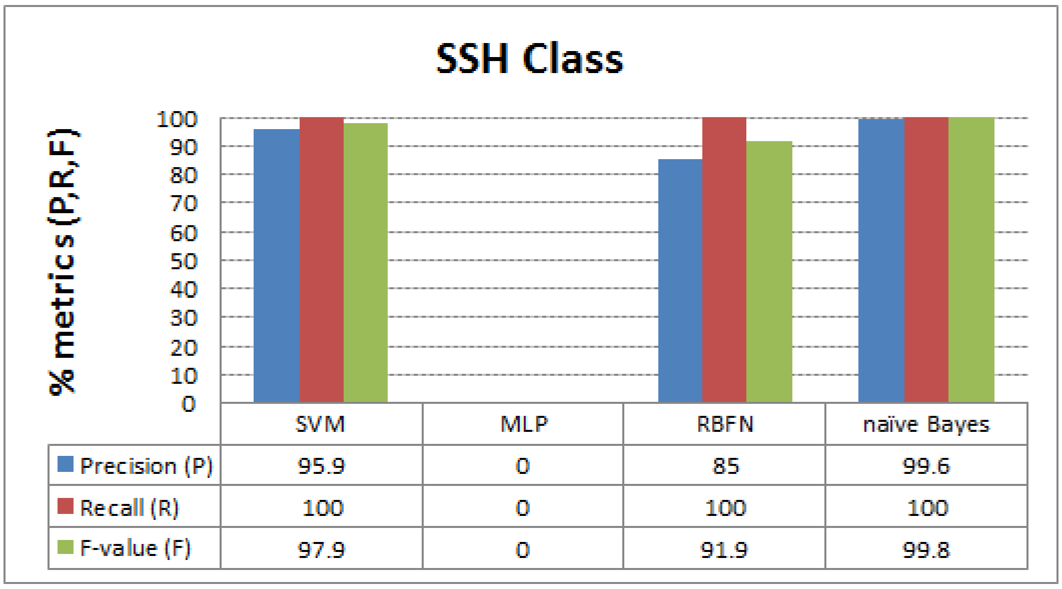
Figure 4.9: L2L Class Performance.

4.5.3 *SSH Class*

The SSH attack type is the second most prevalent attack type on the dataset. This intrusion will also affect the recall value if it is misclassified as normal class.

Another interesting observation as the naïve-Bayes classifier again outperforms the other classifiers in this attack class. It has a precision value of 99.6%, a recall value of 100%, and F-value of 99.8%. It achieved an almost perfect detection rate.

The SVM classifier was the second best with a precision value of 95.9%, a recall value of 100%, and F-value of 97.9%. The RBF network has a precision value of 85%, a recall value of 100%, and F-value of 91.9%. The MLP classifier is not able to classify any of the SSH attacks hence a value of 0% for precision, recall and F-Value.



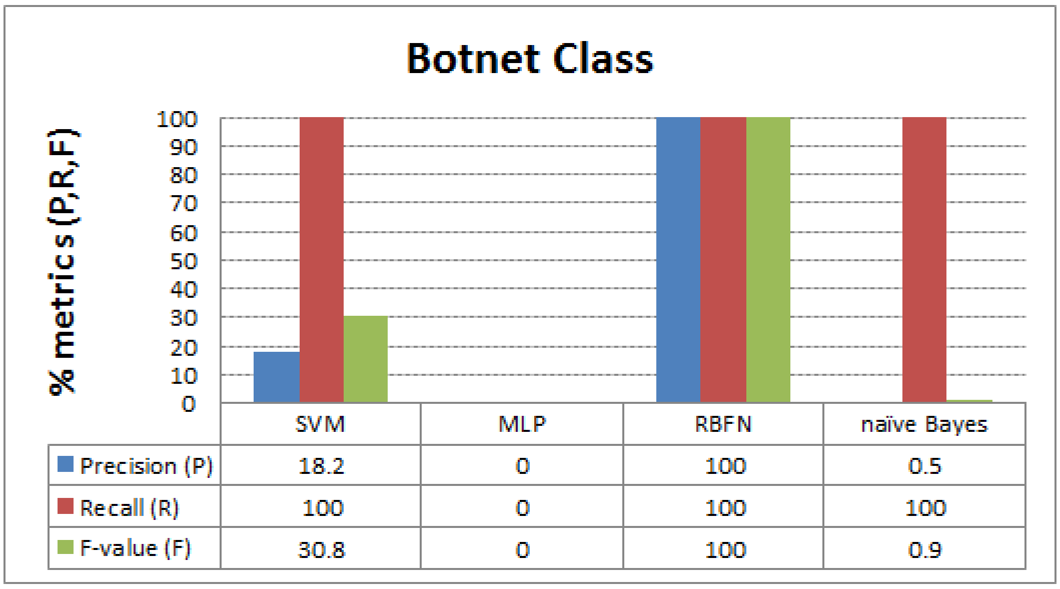
	SVM	MLP	RBFN	naïve Bayes
🟦 Precision (P)	95.9	0	85	99.6
🟥 Recall (R)	100	0	100	100
🟩 F-value (F)	97.9	0	91.9	99.8

Figure 4.10: SSH Class Performance.

The SSH class is the third highest attribute frequency class. The testing and training data contains 3.4% of the SSH class. The poor performance of the MLP classifier can be attributed to this fact.

4.5.4 *Botnet Class*

The Botnet attack type is the least prevalent attack type on the dataset. This intrusion will also affect the recall value if it is misclassified as normal class. This class produces interesting observations of how the classifiers handle low frequency classes. The two instances of this attack are contained in both the training and testing set.



	SVM	MLP	RBFN	naïve Bayes
🟦 Precision (P)	18.2	0	100	0.5
🟥 Recall (R)	100	0	100	100
🟩 F-value (F)	30.8	0	100	0.9

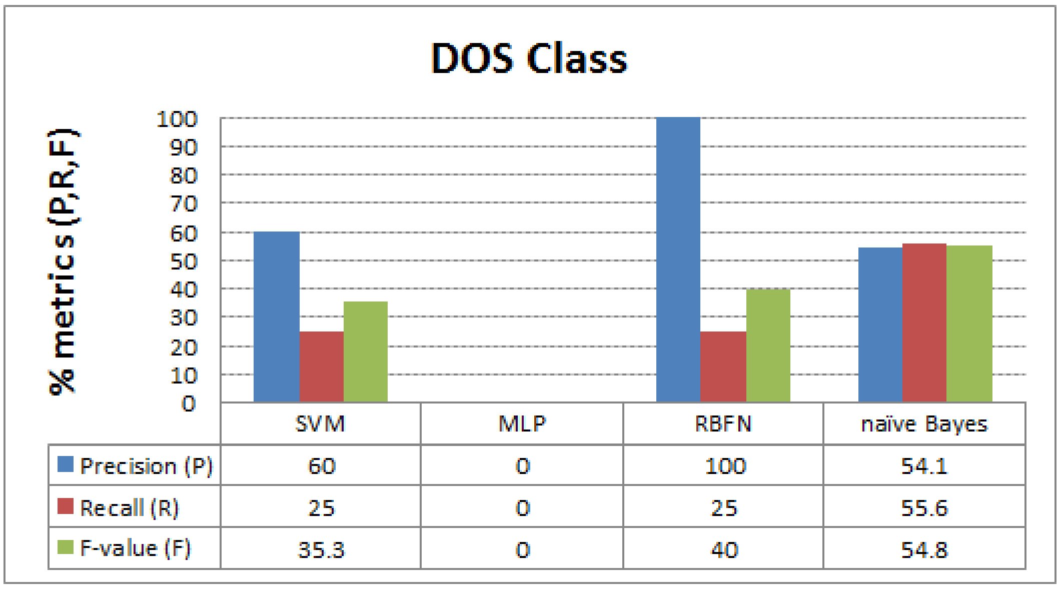
Figure 4.11: Botnet Class Performance.

The RBF network classifier achieved perfect detection of the botnet class. The MLP classifier again is not able to classify the botnet attack. All the other classifiers achieved 100% recall, this might be attributed to the issue of over-fitting because the same attributes are contained in both the training and testing data. The SVM classifier can only manage a precision value of 18.2%, a recall value of 100%, and F-value of 30.8%. The naïve-Bayes classifier only achieves a precision of 0.5% and F-Value of 0.9% as seen on Figure 4.11. The entire data from the seven days that traffic was collected had only 2 instances of the Botnet attack.

4.5.5 *DoS Class*

The DoS attack type is the second least prevalent attack type on the dataset. This intrusion will affect the recall value if it is misclassified as normal class. All the other classes classified as DoS class will also affect the precision of the classifier.

The DoS class is a low frequency class with only 0.2% of the data in both the training and testing data. The naïve-Bayes classifier again outperforms the other classifiers in this attack class as seen on Figure 4.12. Even though it is the better classifier, a precision value of 54.1%, a recall value of 55.6%, and F-value of 54.8% is not good when it comes to intrusion detection. The MLP was not able to classify the DoS attacks. RBF network was the second best classifier in this class with an F-Value of 40%, followed by the SVM with 35.3%. This is the second class that two classifiers outperform the SVM classifier.



	SVM	MLP	RBFN	naïve Bayes
🟦 Precision (P)	60	0	100	54.1
🟥 Recall (R)	25	0	25	55.6
🟩 F-value (F)	35.3	0	40	54.8

Figure 4.12: DoS Class Performance.

Chapter 5

Conclusion and Future Work

5.1 *Conclusion*

In summary, we explored the problem of classification in anomaly network intrusion detection. Anomaly network intrusion detection systems are plagued by high false positive rates which make them impractical to be deployed on live network settings. The high false positive rates are due to misclassification errors. A support vector machine (SVM) classifier is implemented to handle the classification of network attacks and its performance is compared to other classifiers i.e. multilayer perceptrons, naïve-Bayes, and radial basis function network (RBF-N) classifier.

After completing the evaluation of results in Chapter 4, the following conclusions can be drawn. The results show that the SVM classifier is a viable classifier for the problem of classification in anomaly network intrusion detection. We have shown it outperforms multilayer perceptron (MLP), naïve-Bayes, and radial basis function network (RBF-N) in the classification problem of intrusions.

The 99.1% average detection rate and a false positive rate of 4.5% on the testing dataset shows promise for the SVM to be implemented as a classifier of choice for anomaly intrusion detection. The SVM classifier has high detection rates on the SSH and L2L attack types but the successful detection of the DoS and Botnet leave a lot of questions.

It can be concluded that the classifier performance is also affected by the frequency distribution of the attack types in the dataset. Most of the classifiers tested performed poorly on the low attribute frequency classes. This is a worrying trend since some of the low frequency attacks are the most disruptive like DoS and DDoS.

The SVM classifier shows better performance compared to the modified gravitational search algorithm (MGSA) neural network used in [65] with 97.8% detection accuracy and the multi-objective genetic algorithm (MOGA) multilayer perceptron used by [36] with 97% average detection accuracy. The SVM classifier was tested on the ISCX dataset which is the most current dataset that is used in the evaluation of intrusion detection systems. Most of the classifiers tested before used the old outdated KDDCUP'99 dataset. These two are some of the few classifiers that have been evaluated on the ISCX2012 dataset.

5.2 *Future Work*

There are a few ideas that have come up during this work that can be explored in future work.

5.2.1 *Feature Correlation and Dependence*

It would be interesting to explore and find out how some of the individual features do not have an effect on the performance of the classifier but when combined with another feature they greatly affect the classifier performance. This knowledge can further be used in the reduction and selection of features.

5.2.2 *Feature Selection and Reduction*

Most datasets have lots of features and some of these features have little or no impact on the performance of the classifier. For a classifier to be able to be implemented in a live setting of network traffic it should be able to train fast and also analyze the incoming traffic successfully. If a classifier can have almost comparable performance when trained on a reduced feature dataset as in a full feature dataset it might be viable to forgo the performance degradation for speed.

5.2.3 *Incremental Learning*

The idea of incremental learning can be explored. In incremental learning there is no need for retraining the classifier when there is new information available hence updating profiles or signatures dynamically as they are encountered. The incremental approach would probably make the system faster in terms of training as well as testing of the instances. Incremental learning might also improve real time performance.

5.2.4 *Classifier Ensembles*

The idea of combining two or more classifier that have high accuracy performance on specific attack classes can be explored. The evaluation of the results showed that some classifiers performed better than others in the different data classes. The strengths of the different classifiers can be merged together to achieve better classification accuracy for a network anomaly intrusion detection system.

Bibliography

[1] 1998 DARPA Intrusion Detection Evaluation Data Set. Available at `http://www.ll.mit.edu/ideval/data/1998data.html`. [Online; accessed 13- May-2014].

[2] 1999 DARPA Intrusion Detection Evaluation Data Set. Available at `http://www.ll.mit.edu/ideval/data/1998data.html`. [Online; accessed 13- May-2014].

[3] P. Anandan, M. Varma, and J. Joy. Multiple kernel learning. Available at `http://research.microsoft.com/en-us/groups/vgv/`.

[4] D. Anderson, T. Frivold, and A. Valdes. *Next-generation intrusion detection expert system (NIDES): A summary.* SRI International, Computer Science Laboratory, 1995.

[5] J. P. Anderson. Computer security threat monitoring and surveillance. Technical report, Technical report, James P. Anderson Company, Fort Washington, Pennsylvania, 1980.

[6] M. Awad, L. Khan, F. Bastani, and I.-L. Yen. An effective support vector machines (SVMs) performance using hierarchical clustering. In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pages 663–667, Nov 2004.

[7] D. Barbara, W. Ningning, and J. Sushil. Detecting novel network intrusions using Bayes estimators. In *Proceedings of the First SIAM International Conference on Data Mining*, April. 2001.

[8] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011.

[9] W.-H. Chen, S.-H. Hsu, and H.-P. Shen. Application of SVM and ANN for intrusion detection. *Computers & Operations Research*, 32(10):2617 – 2634, 2005. Applications of Neural Networks.

[10] Computer Immune Systems Dataset. Available at `http://www.cs.unm.edu/~immsec/systemcalls.htm`. [Online; accessed 13- May-2014].

[11] C. Cortes and V. Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, Sept. 1995.

[12] J. Cucurull, M. Asplund, and S. Nadjm-Tehrani. Anomaly detection and mitigation for disaster area networks. volume 6307 of *Lecture Notes in Computer Science*, pages 339–359. 2010.

[13] H. Debar, M. Dacier, and A. Wespi. Towards a taxonomy of intrusion-detection systems. *Journal of Graph Theory*, 31(8):805–822, 1999.

[14] D. Denning. An intrusion-detection model. *Journal of Graph Theory*, SE-13(2):222–232, 1987.

[15] O. Depren, M. Topallar, E. Anarim, and M. K. Ciliz. An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. *Expert Systems with Applications*, 29(4):713 – 722, 2005.

[16] J. Dickerson and J. Dickerson. Fuzzy network profiling for intrusion detection. In *Fuzzy Information Processing Society, 2000. NAFIPS. 19th International Conference of the North American*, pages 301 –306, 2000.

[17] T. Gara and C. Warzel. Credit card breach at home depot. `http://www.buzzfeed.com/tomgara/sony-hack`. [Online; accessed 1-April-2015].

[18] A. A. Ghorbani, W. Lu, and M. Tavallaee. *Network Intrusion Detection and Prevention - Concepts and Techniques*, volume 47 of *Advances in Information Security*. Springer, 2010.

[19] A. Ghosh, J. Wanken, and F. Charron. Detecting anomalous and unknown intrusions against programs. In *Computer Security Applications Conference, 1998. Proceedings. 14th Annual*, pages 259–267, Dec 1998.

[20] A. K. Ghosh and A. Schwartzbard. A study in using neural networks for anomaly and misuse detection. In *USENIX Security*, 1999.

[21] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[22] S. S. Haykin, S. S. Haykin, S. S. Haykin, and S. S. Haykin. *Neural networks and learning machines*, volume 3. Pearson Education Upper Saddle River, 2009.

[23] L. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber. A network security monitor. In *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on*, pages 296–304, May 1990.

[24] A. Hernandez. Minecraft data breach affects users. `http://techaeris.com/2015/01/20/reports-minecraft-data-breach-affects-users/`. [Online; accessed 21-April-2015].

[25] J. Hochberg, K. Jackson, C. Stallings, J. McClary, D. DuBois, and J. Ford. Nadir: An automated system for detecting network intrusion and misuse. *Computers & Security*, 12(3):235 –248, 1993.

[26] S.-J. Horng, M.-Y. Su, Y.-H. Chen, T.-W. Kao, R.-J. Chen, J.-L. Lai, and C. D. Perkasa. A novel intrusion detection system based on hierarchical clustering and support vector machines. *Expert Systems with Applications*, 38(1):306 – 313, 2011.

[27] K. Hwang, M. Cai, Y. Chen, and M. Qin. Hybrid intrusion detection with weighted signature generation over anomalous internet episodes. *Dependable and Secure Computing, IEEE Transactions on*, 4(1):41–55, Jan 2007.

[28] K. Ilgun. Ustat: a real-time intrusion detection system for unix. In *Research in Security and Privacy, 1993. Proceedings., 1993 IEEE Computer Society Symposium on*, pages 16–28, May 1993.

[29] Information Exploration Shootout. Available at `http://ivpr.cs.uml.edu/shootout/about.html`. [Online; accessed 13- May-2014].

[30] KDD Cup 1999 Data. Available at `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`. [Online; accessed 21-November-2014].

[31] K. Kendall. *A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems*. PhD thesis, MIT Lincoln Laboratory, 1999.

[32] G. Kim, S. Lee, and S. Kim. A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Systems with Applications*, 41(4, Part 2):1690 – 1700, 2014.

[33] L. Koc, T. A. Mazzuchi, and S. Sarkani. A network intrusion detection system based on a hidden nave Bayes multiclass classifier. *Expert Systems with Applications*, 2012.

[34] B. Krebs. Credit card breach at home depot. `http://krebsonsecurity.com/2014/09/banks-credit-card-breach-at-home-depot/`. [Online; accessed 10-April-2015].

[35] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur. Bayesian event classification for intrusion detection. pages 14 – 23, dec. 2003.

[36] G. Kumar and K. Kumar. A multi-objective genetic algorithm based approach for effective intrusion detection using neural networks. In *Intelligent Methods for Cyber Warfare*, pages 173–200. Springer, 2015.

[37] S. Kumar and E. H. Spafford. A pattern matching model for misuse intrusion detection. 1994.

[38] S. Lee and D. Heinbuch. Training a neural-network based intrusion detector to recognize novel attacks. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 31(4):294–299, Jul 2001.

[39] W. Lee, S. Stolfo, and K. Mok. A data mining framework for building intrusion detection models. In *Security and Privacy, 1999. Proceedings of the 1999 IEEE Symposium on*, pages 120 –132, 1999.

[40] W. Lee and S. J. Stolfo. Data mining approaches for intrusion detection. In *Proceedings of the 7th Conference on USENIX Security Symposium - Volume 7*, SSYM'98, pages 6–6, Berkeley, CA, USA, 1998. USENIX Association.

[41] W. Lee, S. J. Stolfo, and K. W. Mok. Mining audit data to build intrusion detection models. In *KDD*, pages 66–72, 1998.

[42] J. Lei and A. Ghorbani. Network intrusion detection using an improved competitive learning neural network. In *Communication Networks and Services Research, 2004. Proceedings. Second Annual Conference on*, pages 190–197, May 2004.

[43] J. Z. Lei and A. A. Ghorbani. Improved competitive learning neural networks for network intrusion and fraud detection. *Neurocomputing*, 75(1):135 – 145, 2012. Brazilian Symposium on Neural Networks (SBRN 2010) International Conference on Hybrid Artificial Intelligence Systems (HAIS 2010).

[44] K. Leung and C. Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science - Volume 38*, ACSC '05, pages 333–342, 2005.

[45] R. P. Lippmann and R. K. Cunningham. Improving intrusion detection performance using keyword selection and neural networks. *Computer Networks*, 34(4):597 – 603, 2000. Recent Advances in Intrusion Detection Systems.

[46] T. F. Lunt, A. Tamaru, and F. Gillham. *A real-time intrusion-detection expert system (IDES)*. SRI International, Computer Science Laboratory, 1992.

[47] S. Mathew, M. Petropoulos, H. Ngo, and S. Upadhyaya. A data-centric approach to insider attack detection in database systems. volume 6307 of *Lecture Notes in Computer Science*, pages 382–401. 2010.

[48] J. McHugh. Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by lincoln laboratory. *ACM Trans. Inf. Syst. Secur.*, 3(4):262–294, Nov. 2000.

[49] MSS Global Threat Response. Emerging Threat - Anonymous - Operation World Cup/Hacking Cup. `http://www.symantec.com/connect/blogs/emerging-threat-anonymous-operation-world-cuphacking-cup`. [Online; accessed 21-June-2014].

[50] B. Mukherjee, L. Heberlein, and K. Levitt. Network intrusion detection. *Network, IEEE*, 8(3):26–41, 1994.

[51] S. Mukkamala, G. Janoski, and A. Sung. Intrusion detection using neural networks and support vector machines. In *Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on*, volume 2, pages 1702–1707, 2002.

[52] S. Mukkamala, A. Sung, and B. Ribeiro. Model Selection for Kernel Based Intrusion Detection Systems. In *Adaptive and Natural Computing Algorithms*, pages 458–461. 2005.

[53] M. Nassar, R. State, and O. Festor. Monitoring sip traffic using support vector machines. In R. Lippmann, E. Kirda, and A. Trachtenberg, editors, *Recent Advances in Intrusion Detection*, volume 5230 of *Lecture Notes in Computer Science*, pages 311–330. Springer Berlin Heidelberg, 2008.

[54] A. Patcha and J.-M. Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, 51(12):3448 – 3470, 2007.

[55] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23):2435–2463, 1999.

[56] B. Pfahringer. Winning the KDD99 classification cup: Bagged boosting. *SIGKDD Explor. Newsl.*, 1(2):65–66, Jan. 2000.

[57] P. A. Porras and P. G. Neumann. Emerald: Event monitoring enabling responses to anomalous live disturbances. In *In Proceedings of the 20th National Information Systems Security Conference*, pages 353–365, 1997.

[58] R. S. Puttini, Z. Marrakchi, and L. M. A Bayesian classification model for real-time intrusion detection. *AIP Conference Proceedings*, 659(1):150, 2003.

[59] M. Quick, E. Hollowood, C. Miles, and D. Hampson. World's biggest data breaches. `http://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/`. [Online; accessed 30- March-2015].

[60] M. Roesch et al. Snort: Lightweight intrusion detection for networks. In *LISA*, volume 99, pages 229–238, 1999.

[61] M. Salem and S. Stolfo. Modeling user search behavior for masquerade detection. volume 6961 of *Lecture Notes in Computer Science*, pages 181–200. 2011.

[62] S. L. Scott. A Bayesian paradigm for designing intrusion detection systems. *Computational Statistics and Data Analysis*, 45(1):69 – 83, 2004.

[63] M. Sebring, E. Shellhouse, M. Hanna, and R. Whitehurst. Expert systems in intrusion detection: a case study. 1988.

[64] A. A. Sebyala, T. Olukemi, L. Sacks, and D. L. Sacks. Active platform security through intrusion detection using naïve Bayesian network for anomaly detection. In *In: Proceedings of London communications symposium*, 2002.

[65] M. Sheikhan and Z. Jadidi. Flow-based anomaly detection in high-speed links using modified gsa-optimized neural network. *Neural Computing and Applications*, 24(3-4):599–611, 2014.

[66] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers and Security*, 31(3):357 – 374, 2012.

[67] S. Smaha. Haystack: an intrusion detection system. In *Aerospace Computer Security Applications Conference, 1988., Fourth*, pages 37 –44, Dec 1988.

[68] Q. Song, W. Hu, and W. Xie. Robust support vector machine with bullet hole image classification. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 32(4):440–448, Nov 2002.

[69] A. Sung and S. Mukkamala. Identifying important features for intrusion detection using support vector machines and neural networks. In *Applications and the Internet, 2003. Proceedings. 2003 Symposium on*, pages 209–216, Jan 2003.

[70] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. In *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, pages 1 –6, july 2009.

[71] M. Tavallaee, N. Stakhanova, and A. Ghorbani. Toward credible evaluation of anomaly-based intrusion-detection methods. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(5):516–524, Sept 2010.

[72] H. Teng, K. Chen, and S. Lu. Adaptive real-time anomaly detection using inductively generated sequential patterns. In *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on*, pages 278–284, May 1990.

[73] The NSL-KDD Data Set. Available at `http://nsl.cs.unb.ca/NSL-KDD/`. [Online; accessed 13- May-2014].

[74] Unix User Data. Available at `https://archive.ics.uci.edu/ml/machine-learning-databases/UNIX_user_data-mld/`. [Online; accessed 13- May-2014].

[75] H. Vaccaro and G. Liepins. Detection of anomalous computer session activity. In *Security and Privacy, 1989. Proceedings., 1989 IEEE Symposium on*, pages 280–289, May 1989.

[76] G. Vigna and R. Kemmerer. Netstat: a network-based intrusion detection approach. In *Computer Security Applications Conference, 1998. Proceedings. 14th Annual*, pages 25–34, Dec 1998.

[77] G. Wang, J. Hao, J. Ma, and L. Huang. A new approach to intrusion detection using artificial neural networks and fuzzy clustering. *Expert Systems with Applications*, 37(9):6225 – 6232, 2010.

[78] Wikipedia. Kernel machine. `http://commons.wikimedia.org/wiki/File:Kernel_Machine.png`. [Online; accessed 21-Feb-2015].

[79] S. X. Wu and W. Banzhaf. The use of computational intelligence in intrusion detection systems: A review. *Applied Soft Computing*, 10(1):1 – 35, 2010.

[80] N. Ye. *The Handbook of Data Mining.* Human Factors and Ergonomics. CRC Press, 2003.

[81] D.-Y. Yeung and Y. Ding. Host-based intrusion detection using dynamic and static behavioral models. *Pattern Recognition*, 36(1):229 – 243, 2003.

[82] L. Ying, Z. Yan, and O. Yang-jia. The design and implementation of host-based intrusion detection system. In *Intelligent Information Technology and Security Informatics (IITSI), 2010 Third International Symposium on*, pages 595 –598, April 2010.

[83] C. Zhang, J. Jiang, and M. Kamel. Intrusion detection using hierarchical neural networks. *Pattern Recognition Letters*, 26(6):779 – 791, 2005.

[84] J. Zhang and M. Zulkernine. A hybrid network intrusion detection technique using random forests. In *Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on*, pages 8 pp.–, April 2006.

Appendix A

Data Breaches

A.1 Stolen Data Black Market Prize

Form of Data	**Black Market Price,(USD)**
CVV	2
Credit card, stale data	2-7
Full personal information	3
Bank account details	5
Health credentials	10
Credit card, market flooded	10-12
PayPal / eBay account	27
Credit card, freshly acquired	20-45
Spam email list	100
"Executive" credit card	8000
Zero-day	up to 250,000

Table A.1: Black Market Data Price [59]

A.2 Data Breaches

ENTITY	YEAR	ORG	METHOD OF LEAK	STOLEN RECORDS	DS[1]
Australian Immigration Department	2015	government	accidentally published	500000	5
British Airways	2015	retail	hacked	500000	1
Slack	2015	tech	poor security	500000	1
Twitch.tv	2015	health	hacked	10000000	1
Premera	2015	health	hacked	11000000	5
Uber	2015	tech	poor security	50000	1
Anthem	2015	healthcare	hacked	80000000	20
Sony Pictures	2014	media	hacked	10000000	20
JP Morgan Chase	2014	financial	hacked	76000000	300
Gmail	2014	web	hacked	5000000	1
Home Depot	2014	retail	hacked	56000000	300
Mozilla	2014	web	poor security	760000	20
Community Health Services	2014	healthcare	hacked	4500000	20
Dominios Pizzas (France)	2014	web	hacked	600000	1
LexisNexis	2014	tech	hacked	1000000	300
AOL	2014	web	hacked	24000000	1
Korea Credit Bureau	2014	financial	inside job	20000000	50000
Target	2014	retail	hacked	70000000	200
Ebay	2014	web	hacked	145000000	1
Adobe	2014	tech	hacked	152000000	50000
Neiman Marcus	2014	retail	hacked	1100100	20

Table A.2: Data Breaches [59]

[1] 1: Just email address/Online information, 20: SSN/Personal details, 300: Credit card information, 4000: Email password/Health records, 50000: Full bank account details.

A.3 Dataset Feature Description

	Feature Name	Feature Description
1	appName	Name of application used
2	totalSourceBytes	Amount of total bytes sent from the source
3	totalDestinationBytes	Amount of total bytes received at the destination
4	totalDestinationPackets	Amount of total packets received at the destination
5	totalSourcePackets	Amount of total packets sent from the source
6	sourcePayloadAsBase64	Payload from the source in Base64 character encoding
7	sourcePayloadAsUTF	Payload from the source in UTF character encoding
8	destinationPayloadAsBase64	Payload at the destination in Base64 character encoding
9	destinationPayloadAsUTF	Payload at the destination in UTF character encoding
10	direction	Direction of the flow e.g remote to local
11	sourceTCPFlagsDescription	Action TCP source requests e.g Push, Synchronize, Finish, Acknowledge, Reset
12	destinationTCPFlagsDescription	Action TCP destination performs
13	source	Source IP address
14	protocolName	Protocol used by the flow e.g TCP-IP
15	sourcePort	Port number of the source
16	destination	Destination IP address
17	destinationPort	Port number of the destination
18	startDateTime	Time flow starts
19	stopDateTime	Time flow stops
20	Tag	Label of flow as either normal or attack

Table A.3: Data Description